Installing Development-Environments

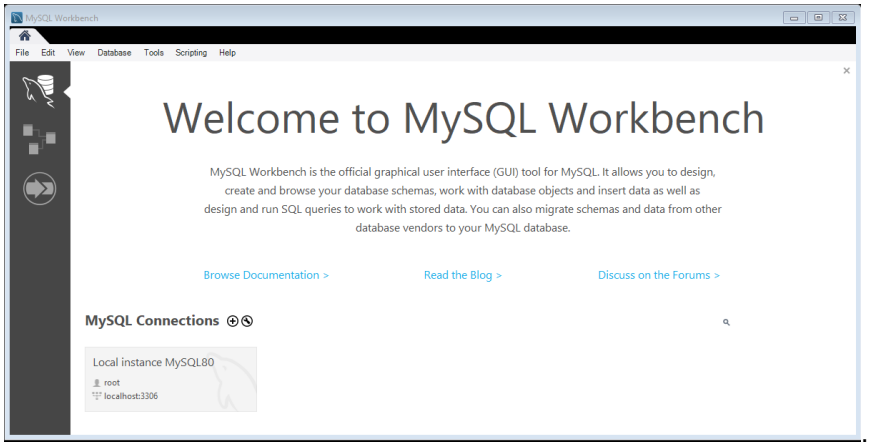
J2EE, MySQL,Phyton,Perl

Last Changes: Walter Rothlin / Dec 25th 2019 / 09:00

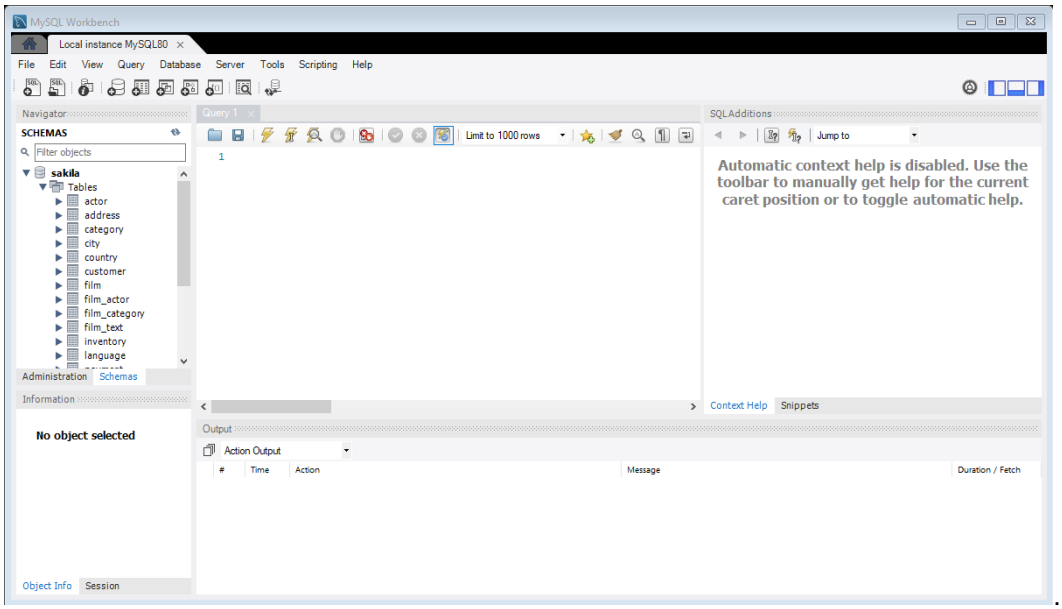
Index

Dowloads 5
Java Setup 5
Download JDK (Java Development Kit) 5
Install JDK 6
Glassfish Server Setup 9
Download Glassfish 9
Install Glassfish 9
Eclipse for JEE Developers Setup 10
Download Eclipse 10
First Start 12
Install JDK in Eclipse 13
Add Glassfish Tools for Eclipse 16
Setup Glassfish with Eclipse 17
Start Glassfish Admin Console 22
Deploy whenever you press CTRL-S to your source file 24
Testing JEE installation 25
Test the Installation: Create first Servlet and JSP, deploy and run it 25
First Servlet Code 29
Testing FirstServlet.java 30
JEE development 31
EJB (stateless and stateful) gets called from Servlet / JSP (in same Web-App) 31
EJB (stateless and stateful) gets called from **Servlet** remotely 42
EJB (stateless and stateful) gets called from a **Java-Client (POJO)** 48
Create Bean to get called from Servlet / JSP in the "FirsServletProject" 54
Call Bean-method from Servlet (same web-App): 56
Testing Servlet calling local bean 57
Call Bean-method from JSP (same web-App) 58
Testing JSP calling EJB local 58
Create an EJB (Enterprise Java Bean) in another project (Remote) 59
First EJB Code 61
Deploy EJB 62
Call EJB Methode from "FirstServlet" 63
Fixen der Projektübergreifenden Interfaces 64
Testen des Servlets mit EJB local and remote call 64

Java-client calls EJB-Methode 65
Create own WEB-Service 71
Create Java-Client calling WEB-Service 74
Create a Java Client which calls a public Web-Service 78
Create TCP/IP Server-Client 78
Create TCP/IP Server-Client advanced 81
Create RMI Client-Server 84
Troubleshooting "DeploymentException" on Glassfish startup 92
Python 95
PyCharm von JetBrains 97
MySql (Workbench 8.0.18) 99
https://dev.mysql.com/downloads/workbench/ 99
Start workbench 105



..... 105



..... 105
Create local User in MySQL Workbench 106
Test MySQL 110
Give a Remote Computer access to MySQL 112
Download MySQL JDBC driver 116

Create a new MySQL Connection 118
Create DB-Connection with DBeaver 120
Basic MySql commands 126
Setup a DB Connection via JNDI on the Application Server 127
Javadoc 131
Generating Javadoc 131
Junit 132
Create Junit Test 132
Coverage - Install EclEmma Plugin 133
Coverage - Run Test with coverage report 133
Common Help 134
Finden der aktuelle IP eines Computers 134
Helpful Links for EJB / Web-Service 135
Tips and Tricks for Eclipse 136
Obsolete: Download and Install MySQL Eclipse Plugin – Toad extension 136
Add MySQL JDBC driver 137
Add MySQL connection 139
New jar to eclipse project 141
Shortcuts in Eclipse 142
Automated publishing when source has been saved 143
Import Project in eclipse 144

Dowloads

This is the list of package downloads

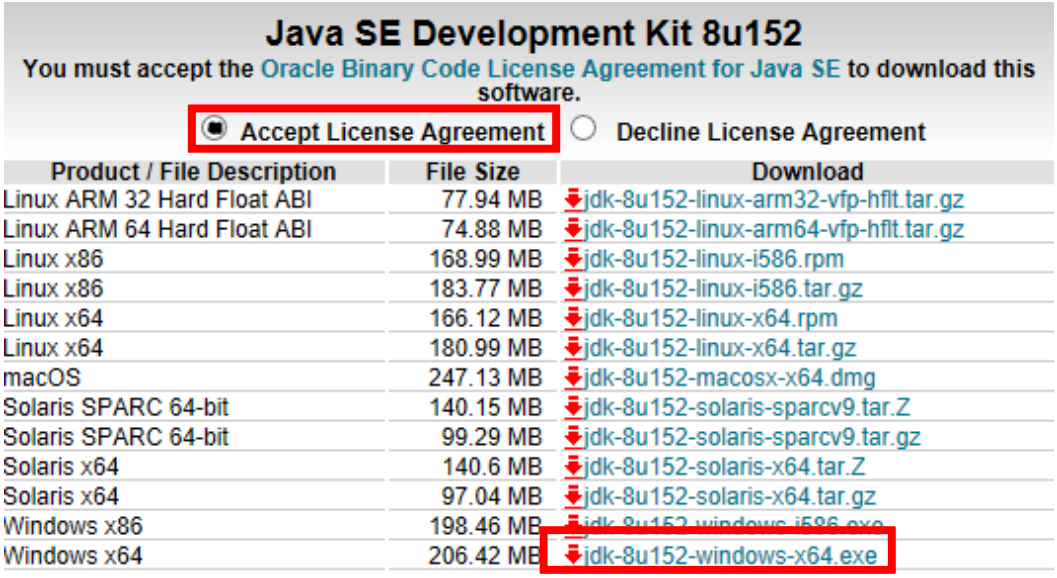
- Java JDK: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Glassfish: https://javaee.github.io/glassfish/download
- Eclipse: https://www.eclipse.org/downloads/eclipse-packages/
- Visual C++: https://www.microsoft.com/en-US/download/details.aspx?id=48145.
- MySQL: https://downloads.mysql.com/archives/get/file/mysql-installer-community-5.7.10.0.msi
- JDBC Connector: https://dev.mysql.com/downloads/connector/j/

Java Setup

Download JDK (Java Development Kit)

Download the latest JDK from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html(January *2018: version 8u152)*

Accept their License Agreement after carefully reading it by selecting the left radio-button. Choose the Windows version from the List and download it.



Java SE Development Kit 8u152

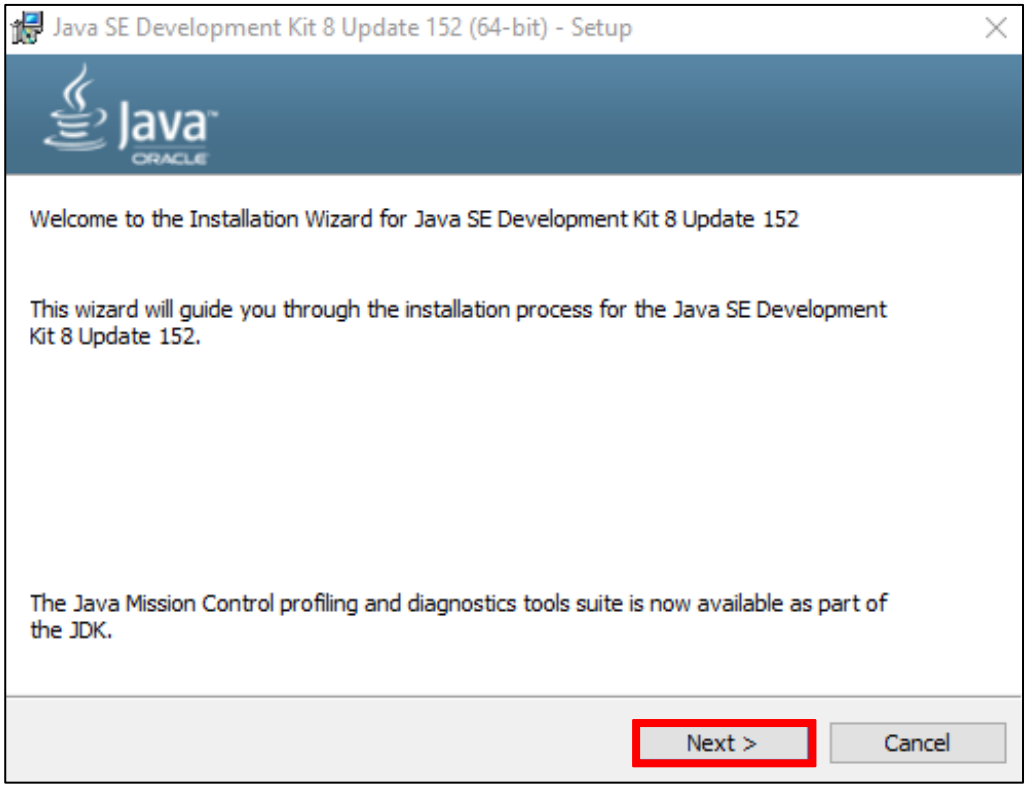
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

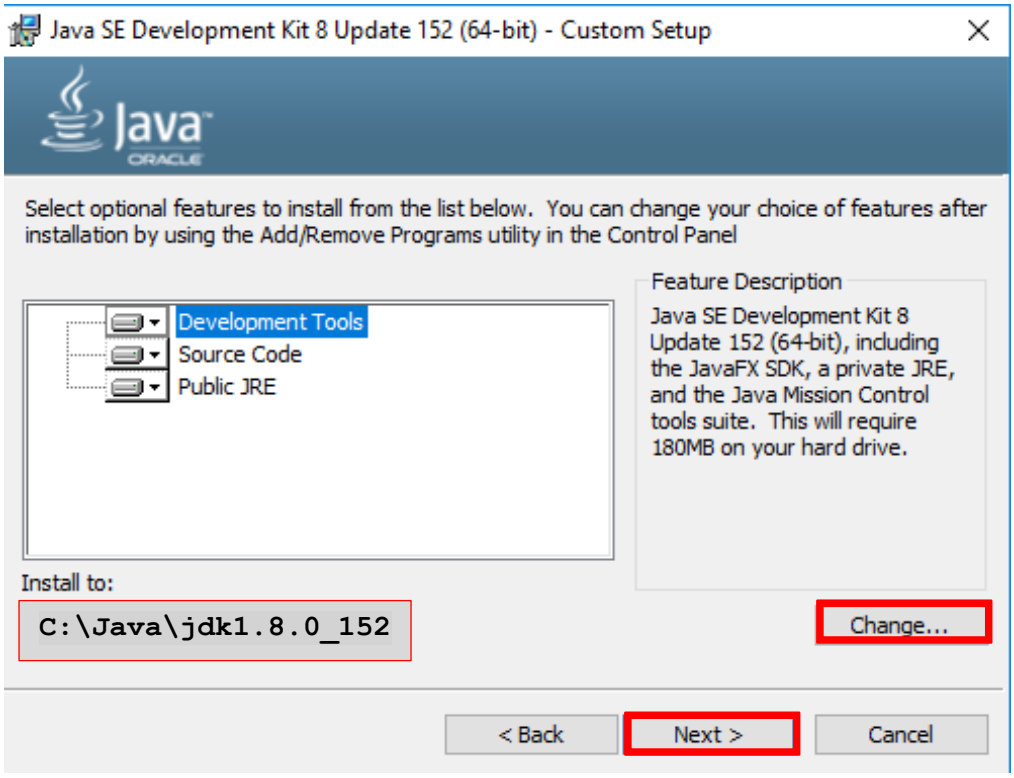
Accept License Agreement Decline License Agreement

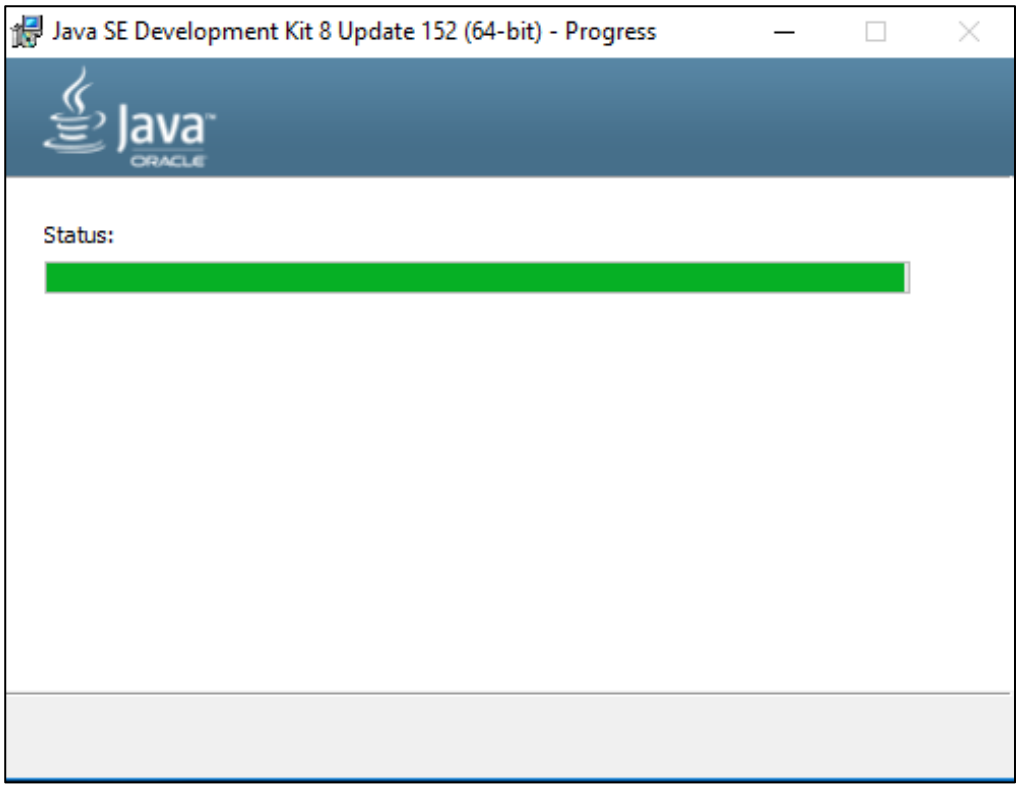
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.94 MB	jdk-8u152-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u152-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.99 MB	jdk-8u152-linux-i586.rpm
Linux x86	183.77 MB	jdk-8u152-linux-i586.tar.gz
Linux x64	166.12 MB	jdk-8u152-linux-x64.rpm
Linux x64	180.99 MB	jdk-8u152-linux-x64.tar.gz
macOS	247.13 MB	jdk-8u152-macosx-x64.dmg
Solaris SPARC 64-bit	140.15 MB	jdk-8u152-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u152-solaris-sparcv9.tar.gz
Solaris x64	140.6 MB	jdk-8u152-solaris-x64.tar.Z
Solaris x64	97.04 MB	jdk-8u152-solaris-x64.tar.gz
Windows x86	198.46 MB	jdk-8u152-windows-i586.exe
Windows x64	206.42 MB	jdk-8u152-windows-x64.exe

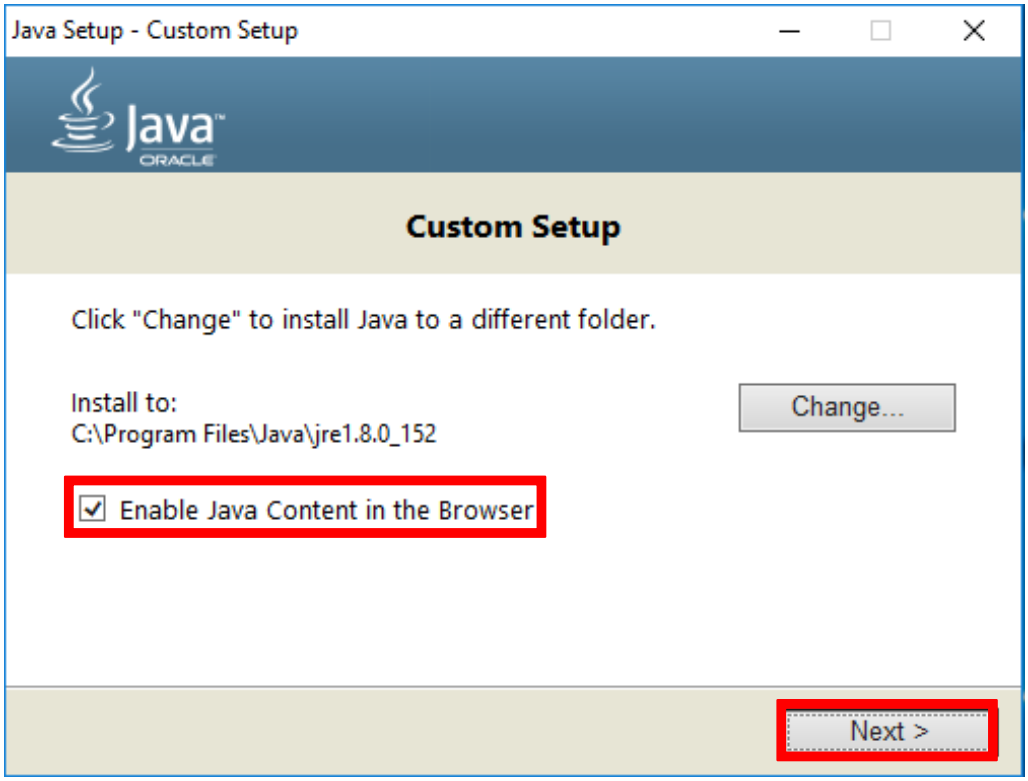
Install JDK

Run the previously downloaded file and follow the instructions. The JDK includes the JRE so you have to specify both installation paths. Both of them shouldn't be changed so they're installed to `C:\Java\jdk1.8.0_152`.






Java SE Development Kit 8 Update 152 (64-bit) - Progress
Java
ORACLE
Status:


Java Setup - Custom Setup
Java
ORACLE
Custom Setup
Click "Change" to install Java to a different folder.
Install to:
C:\Program Files\Java\jre1.8.0_152
Change...
Enable Java Content in the Browser
Next >

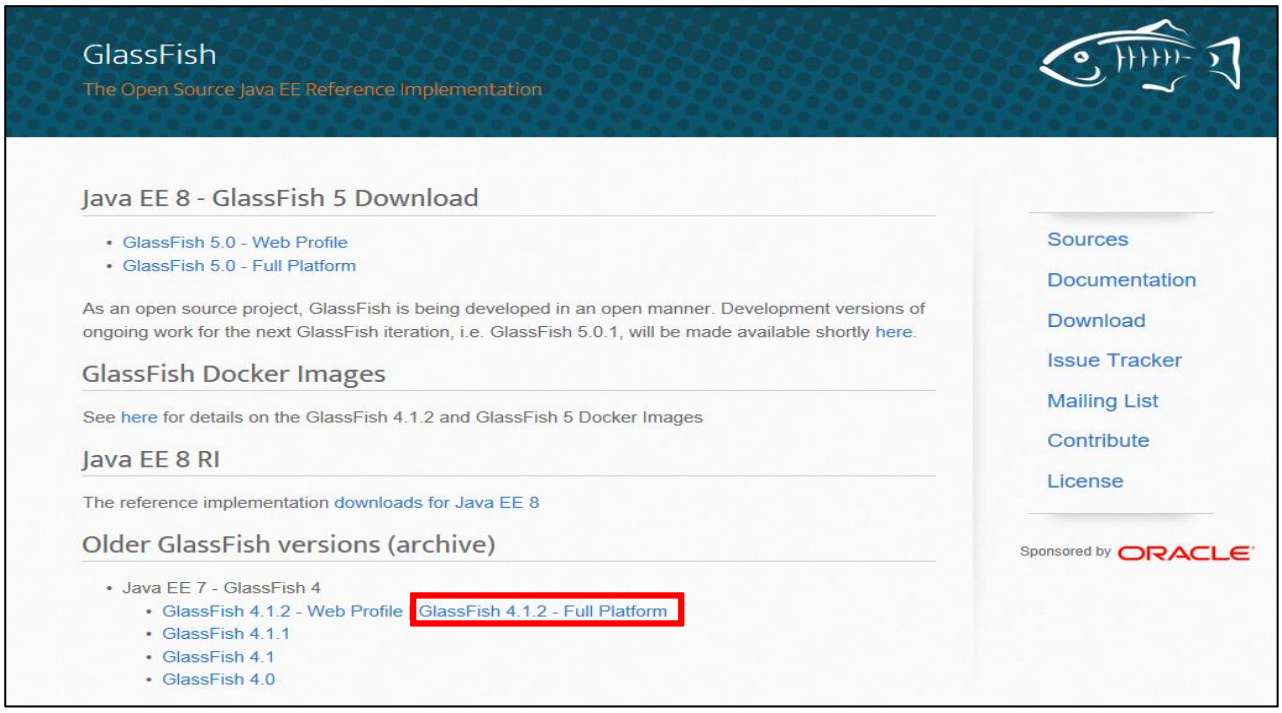
Java Setup - Progress
Java
ORACLE
Status: Installing Java
3 Billion
Devices Run Java
Java
#1 Development Platform
ORACLE

Java SE Development Kit 8 Update 152 (64-bit) - Complete
Java
ORACLE
Java SE Development Kit 8 Update 152 (64-bit) Successfully Installed
Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.
Next Steps
Close

Glassfish Server Setup

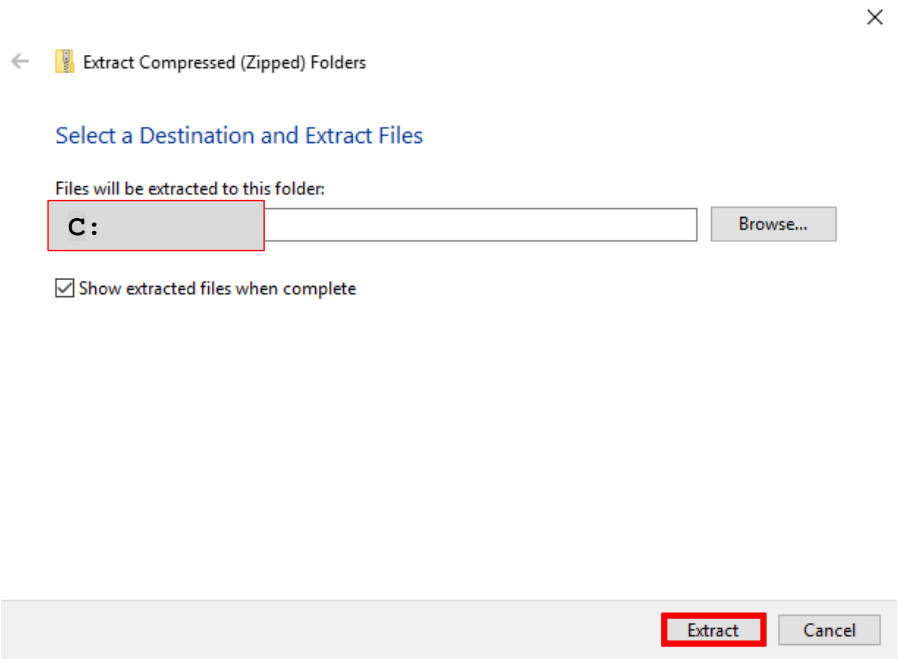
Download Glassfish

Download the latest Full Platform Server from https://javaee.github.io/glassfish/download.
(January 2018: version 4.1.2)



Install Glassfish

Extract the compressed zip-file to `C:` (it gets extracted `to C:\glassfish4`).



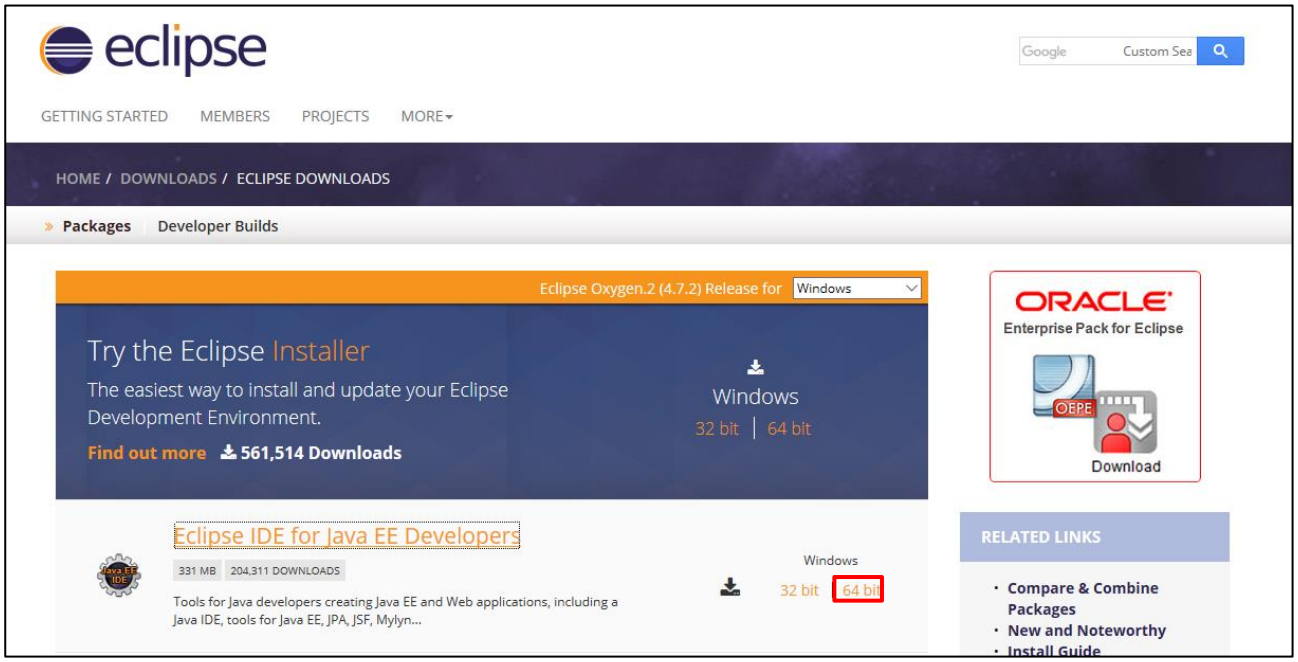
Further Steps to Setup the Glassfish Server will be taken after eclipse is installed (next chapter).

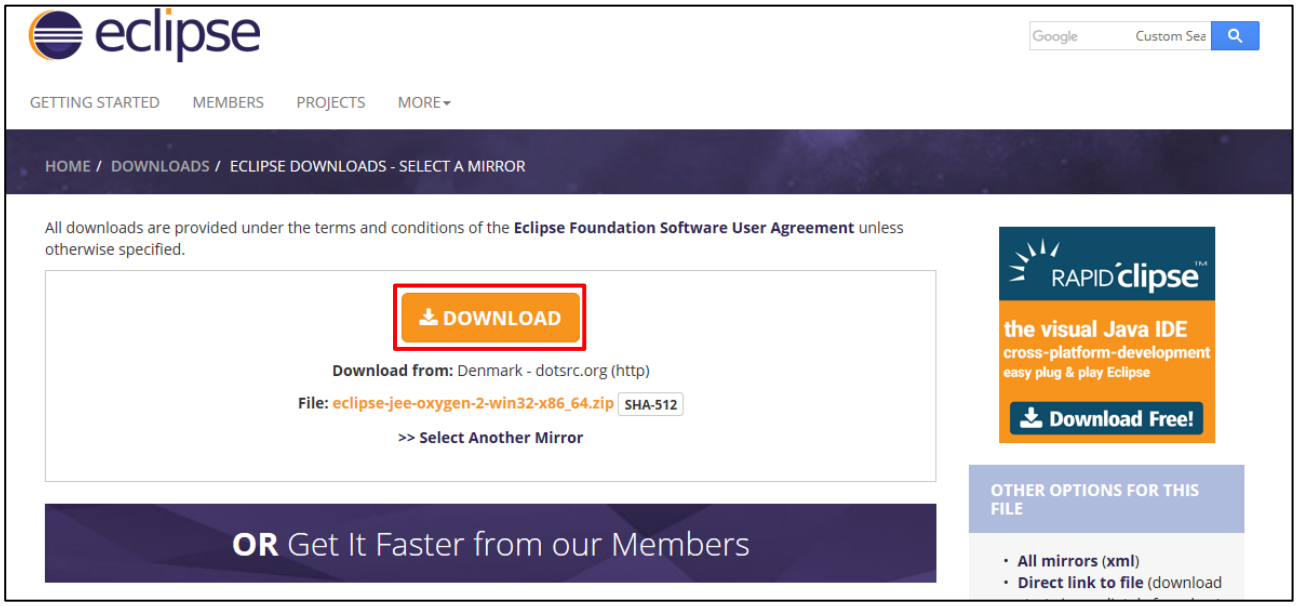
Eclipse for JEE Developers Setup

Download Eclipse

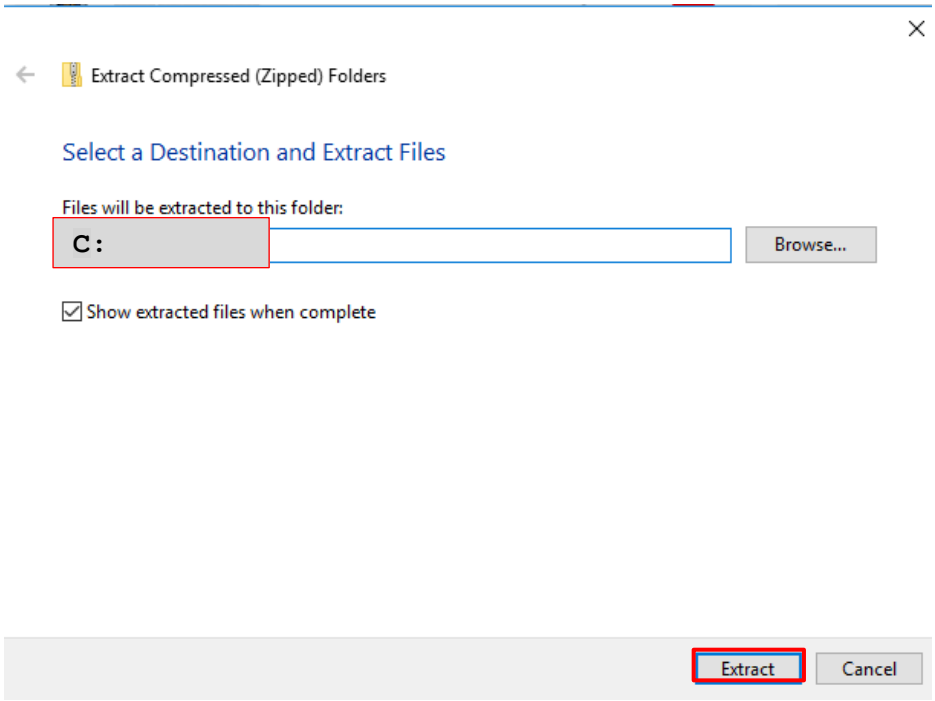
Download the latest 64-bit version of Eclipse for JEE from https://www.eclipse.org/downloads/eclipse-packages/.

(January 2018: Eclipse Oxygen)

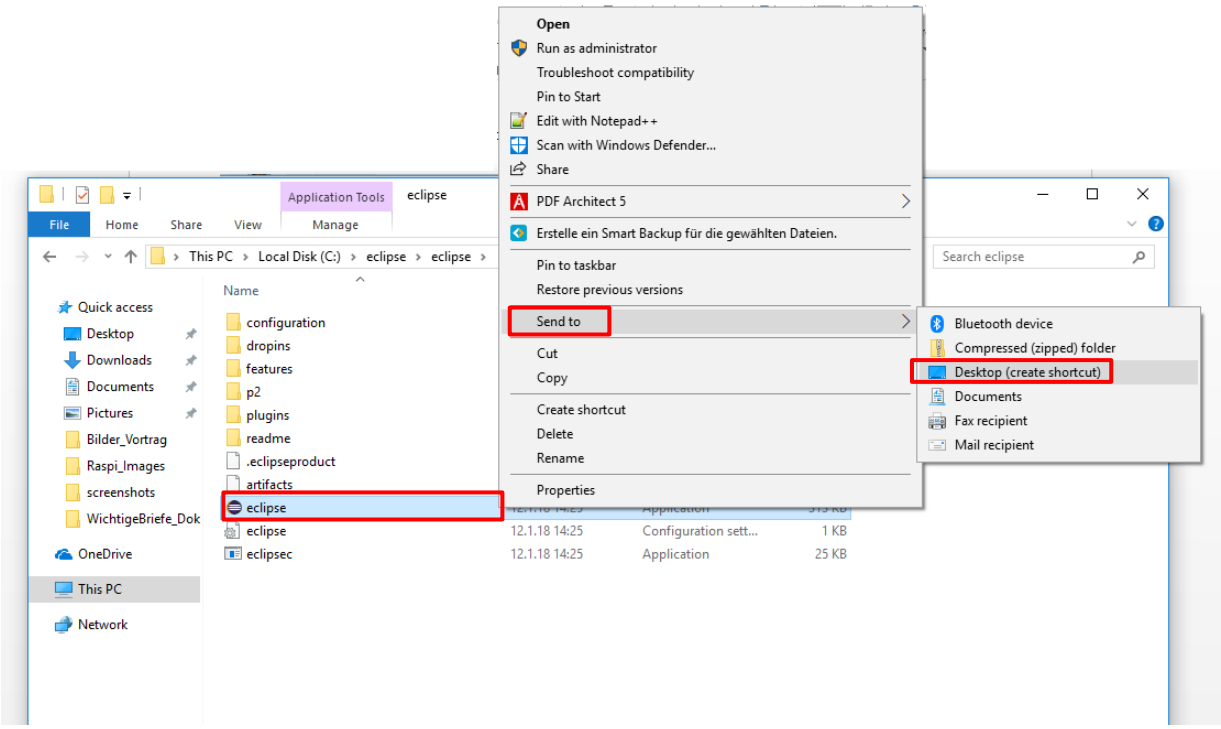




Then simply unzip in C: (it gets extracted to C:\eclipse) and run Eclipse.

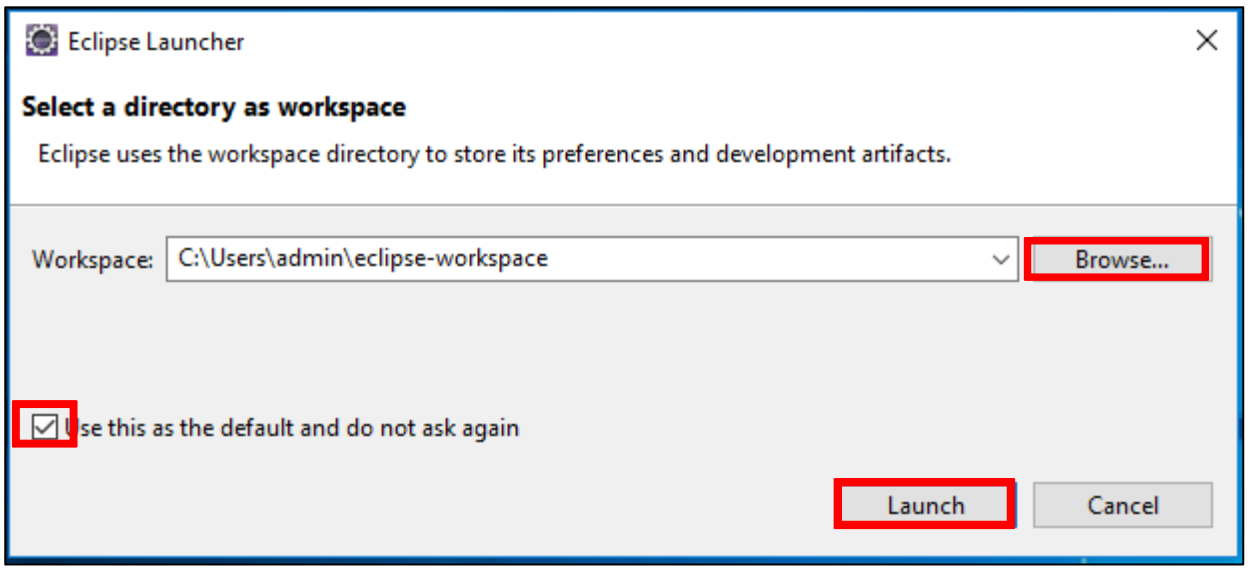


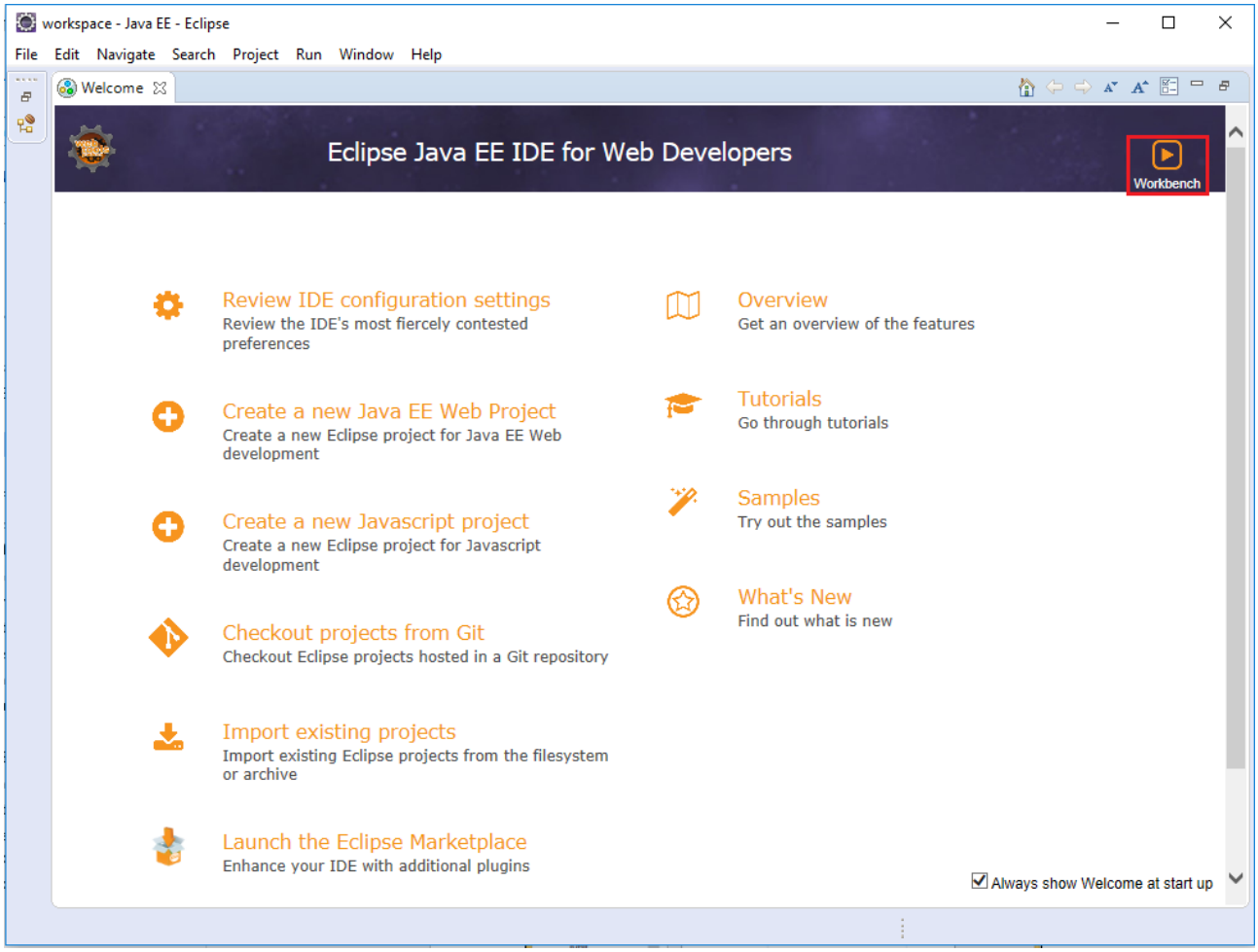
For convenience: **`Right click`** `eclipse.exe -> send to -> desktop (create shortcut)`



First Start

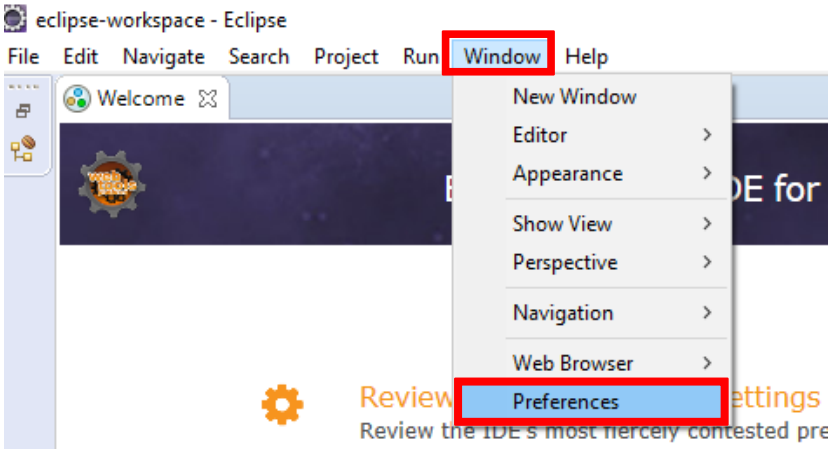
You can select now any directory you want, but to keep it simple just leave it how it is.
Tick "use this as default and do not ask again" and press "OK".

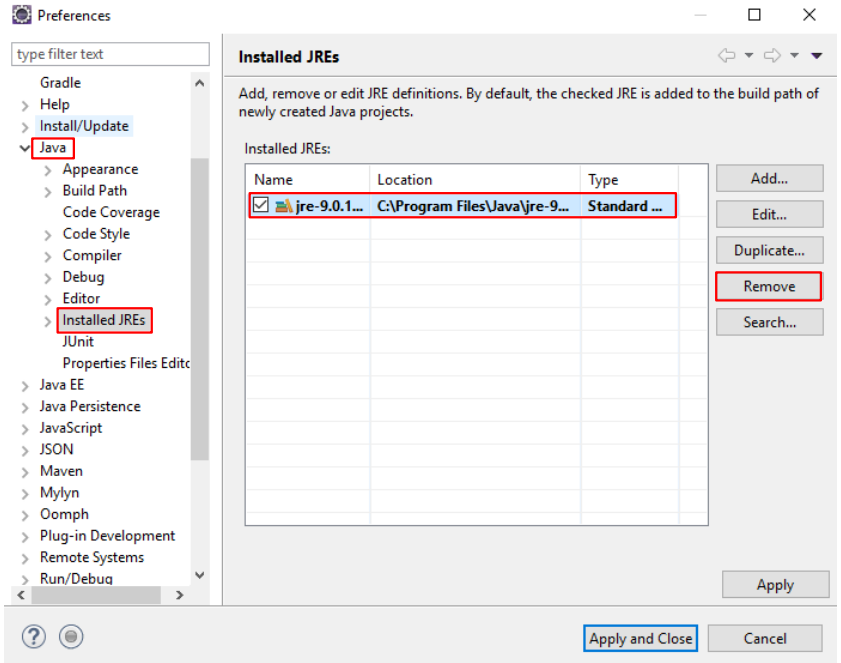


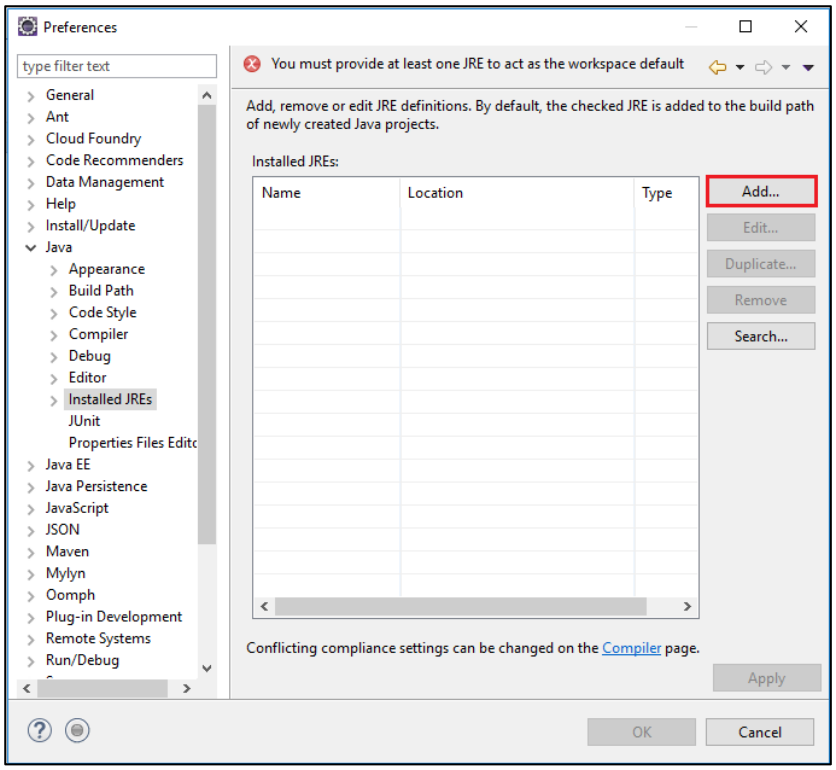


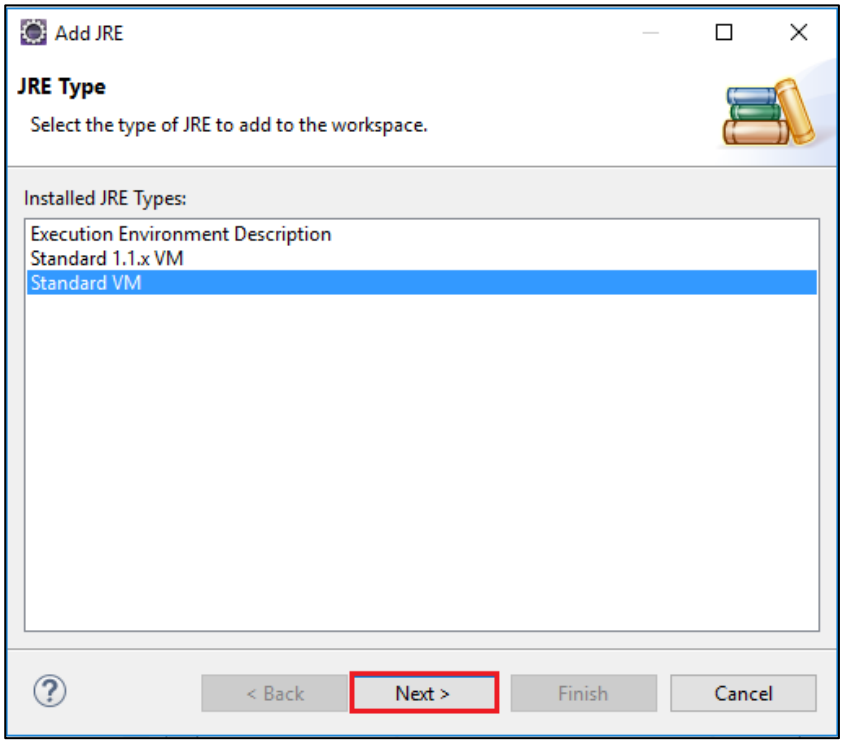
Install JDK in Eclipse

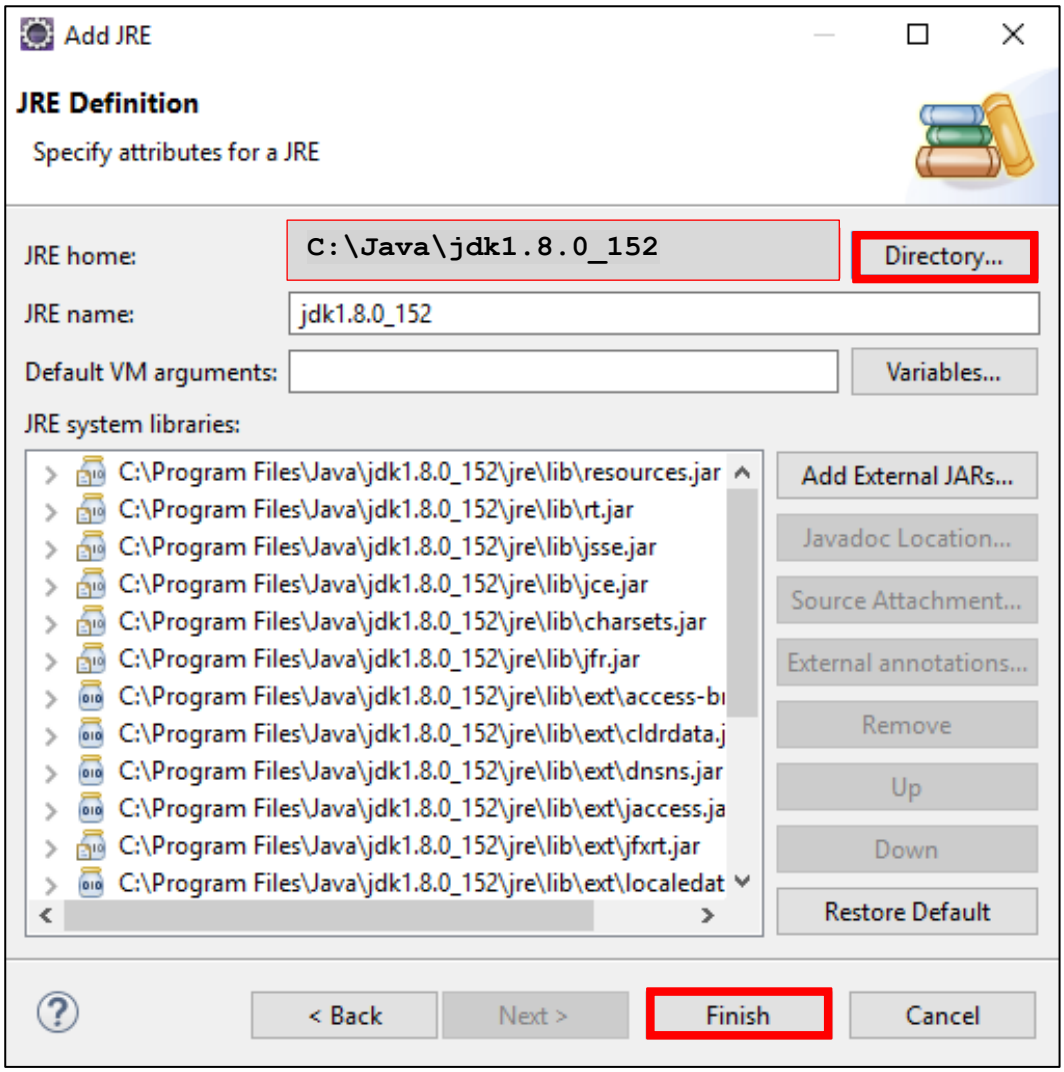
`Window -> Preferences -> Java -> Installed JREs`

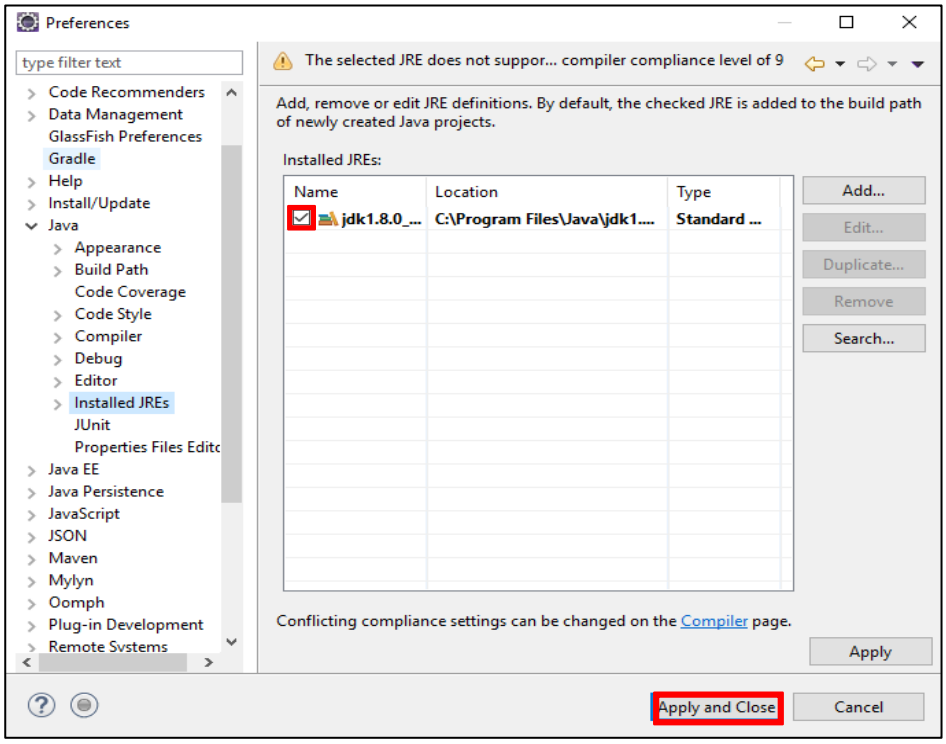





Preferences
type filter text
General
Ant
Cloud Foundry
Code Recommenders
Data Management
Help
Install/Update
Java
Appearance
Build Path
Code Style
Compiler
Debug
Editor
Installed JREs
JUnit
Properties Files Editc
Java EE
Java Persistence
JavaScript
JSON
Maven
Mylyn
Oomph
Plug-in Development
Remote Systems
Run/Debug
You must provide at least one JRE to act as the workspace default
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
Type
Add...
Edit...
Duplicate...
Remove
Search...
Conflicting compliance settings can be changed on the Compiler page.
Apply
OK
Cancel

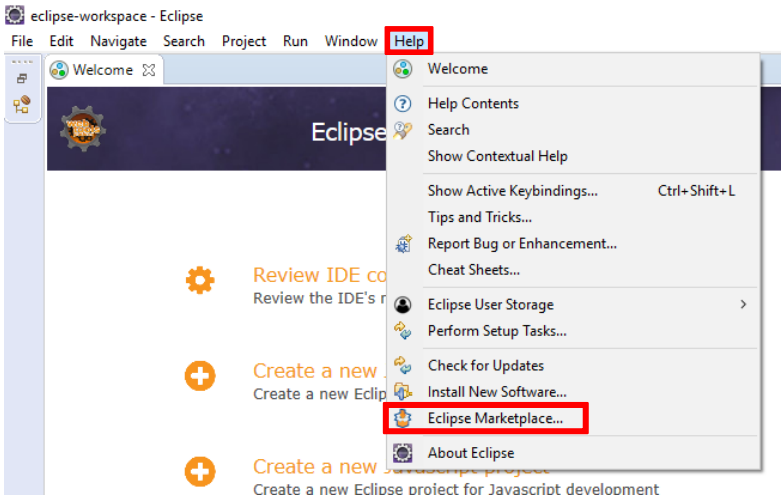

Add JRE
JRE Type
Select the type of JRE to add to the workspace.
Installed JRE Types:
Execution Environment Description
Standard 1.1.x VM
Standard VM
< Back
Next >
Finish
Cancel

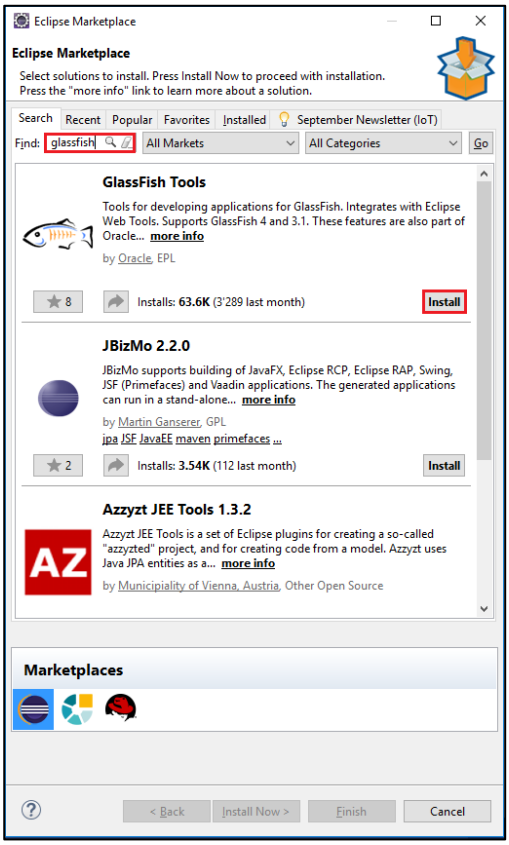

Add JRE
JRE Definition
Specify attributes for a JRE
JRE home:
C:\Java\jdk1.8.0_152
Directory...
JRE name:
jdk1.8.0_152
Default VM arguments:
Variables...
JRE system libraries:
C:\Program Files\Java\jdk1.8.0_152\jre\lib\resources.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\rt.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\jce.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\access-b
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\cldrdata.j
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\jaccess.ja
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\jfxrt.jar
C:\Program Files\Java\jdk1.8.0_152\jre\lib\ext\localedat
Add External JARs...
Javadoc Location...
Source Attachment...
External annotations...
Remove
Up
Down
Restore Default
< Back
Next >
Finish
Cancel

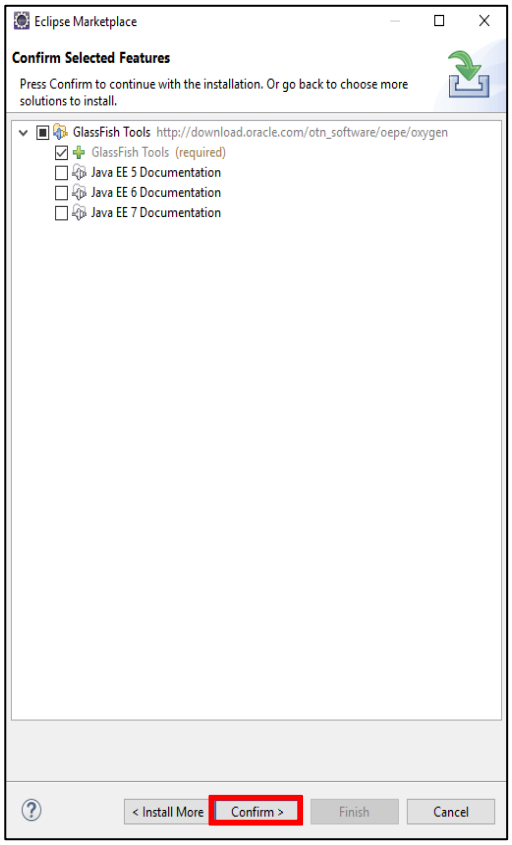

Preferences
type filter text
Code Recommenders
Data Management
GlassFish Preferences
Gradle
Help
Install/Update
Java
Appearance
Build Path
Code Coverage
Code Style
Compiler
Debug
Editor
Installed JREs
JUnit
Properties Files Edit
Java EE
Java Persistence
JavaScript
JSON
Maven
Mylyn
Oomph
Plug-in Development
Remote Systems
The selected JRE does not suppor... compiler compliance level of 9
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
Type
jdk1.8.0_...
C:\Program Files\Java\jdk1....
Standard ...
Add...
Edit...
Duplicate...
Remove
Search...
Conflicting compliance settings can be changed on the Compiler page.
Apply
Apply and Close
Cancel

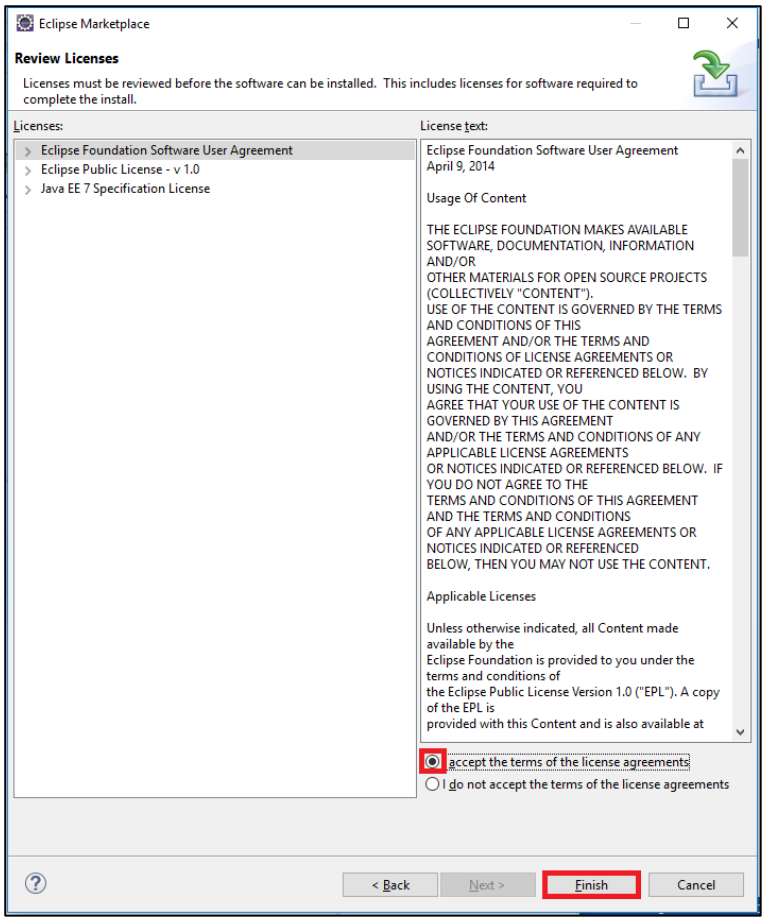
Add Glassfish Tools for Eclipse

`Help -> Eclipse Marketplace -> search for "glassfish"`



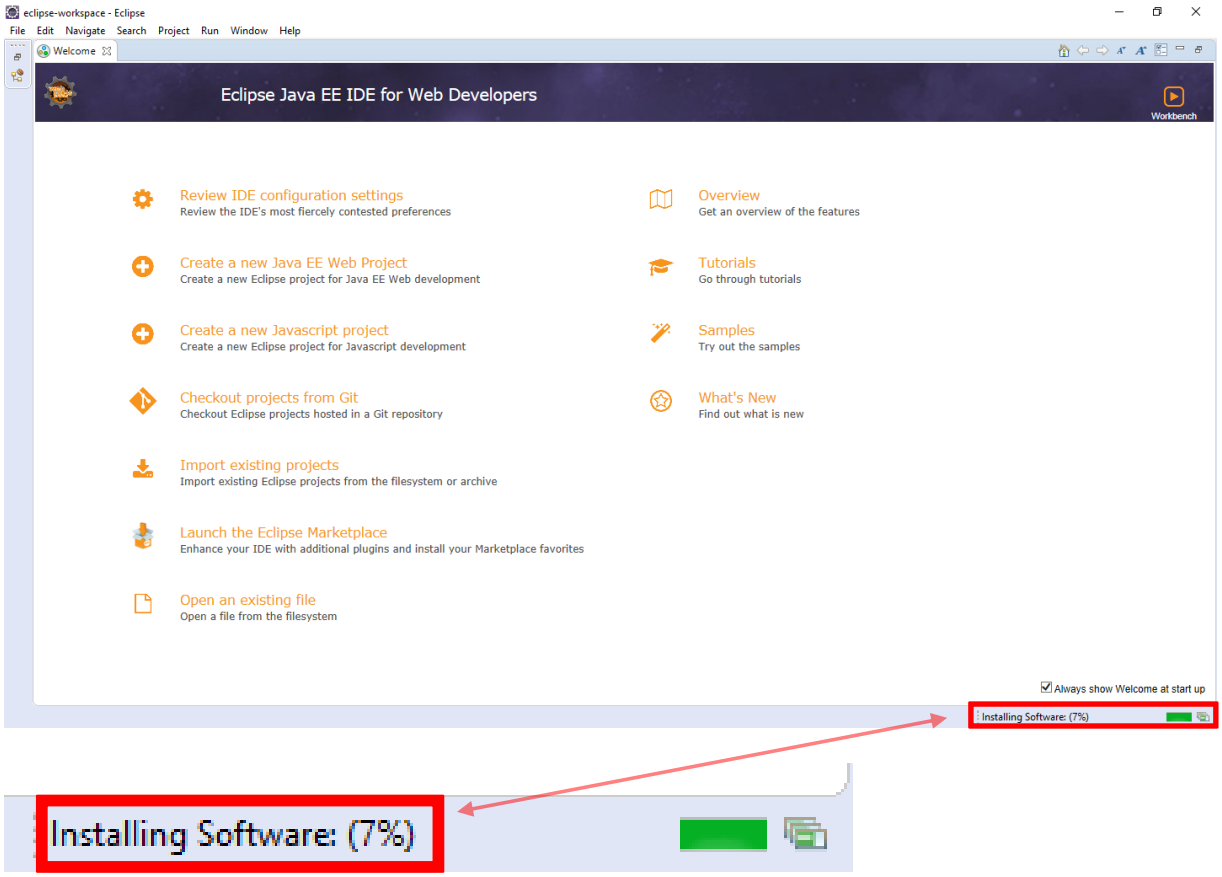




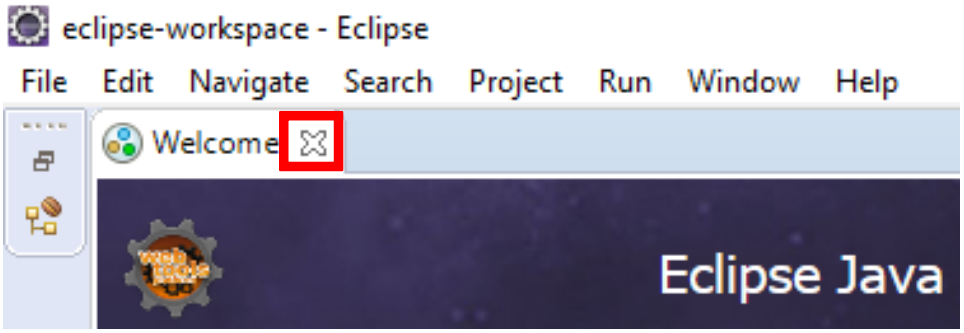


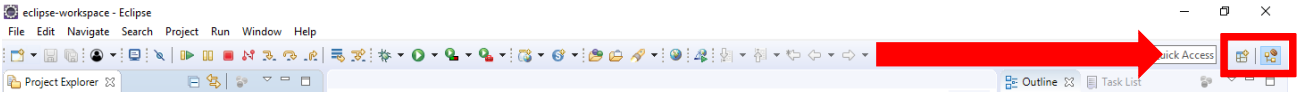
Setup Glassfish with Eclipse

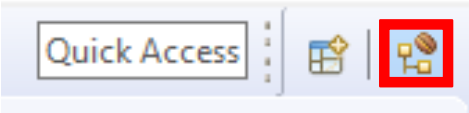
After the software has finished installing restart Eclipse.



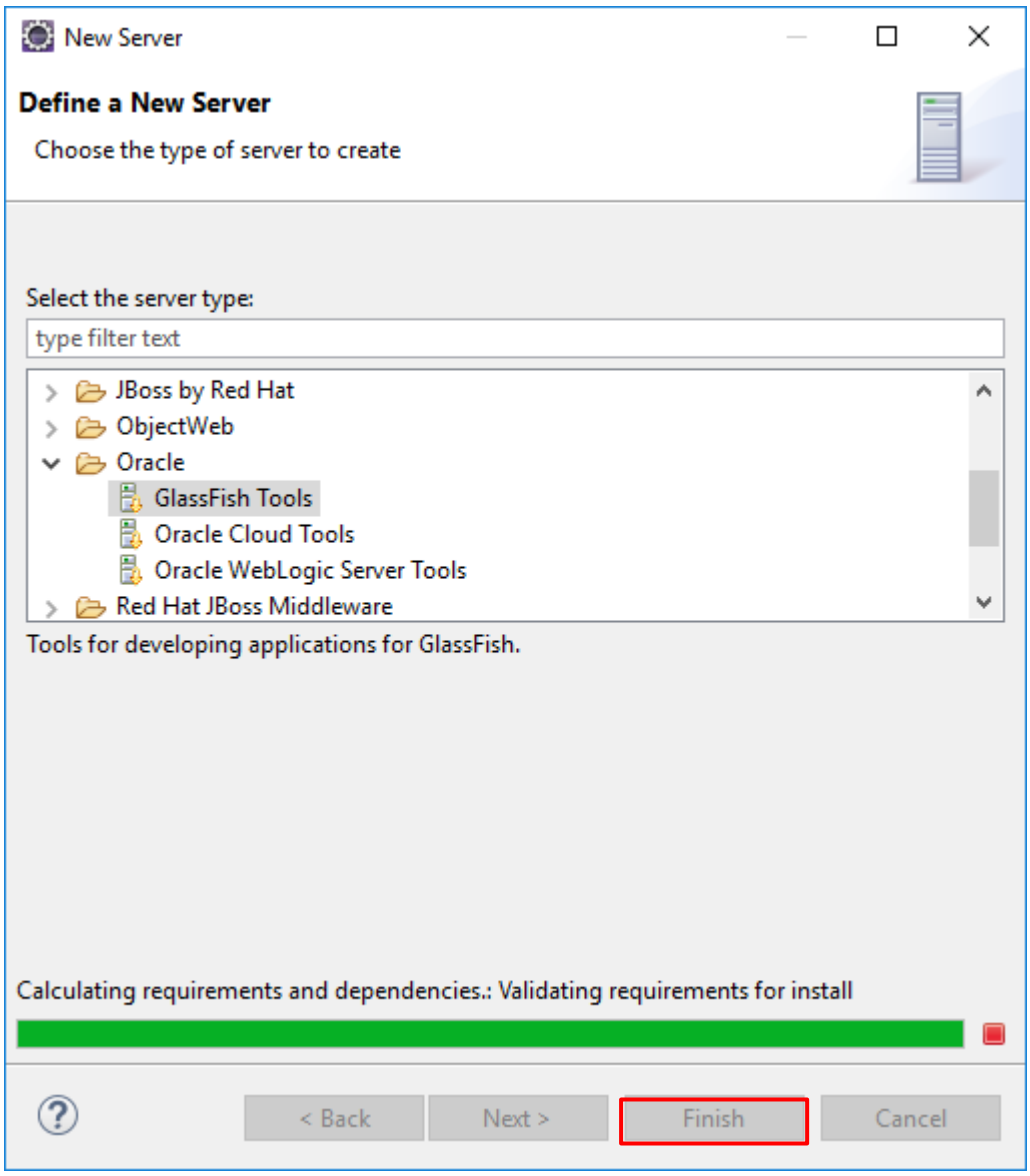
Change your view in the top right corner to Java EE.



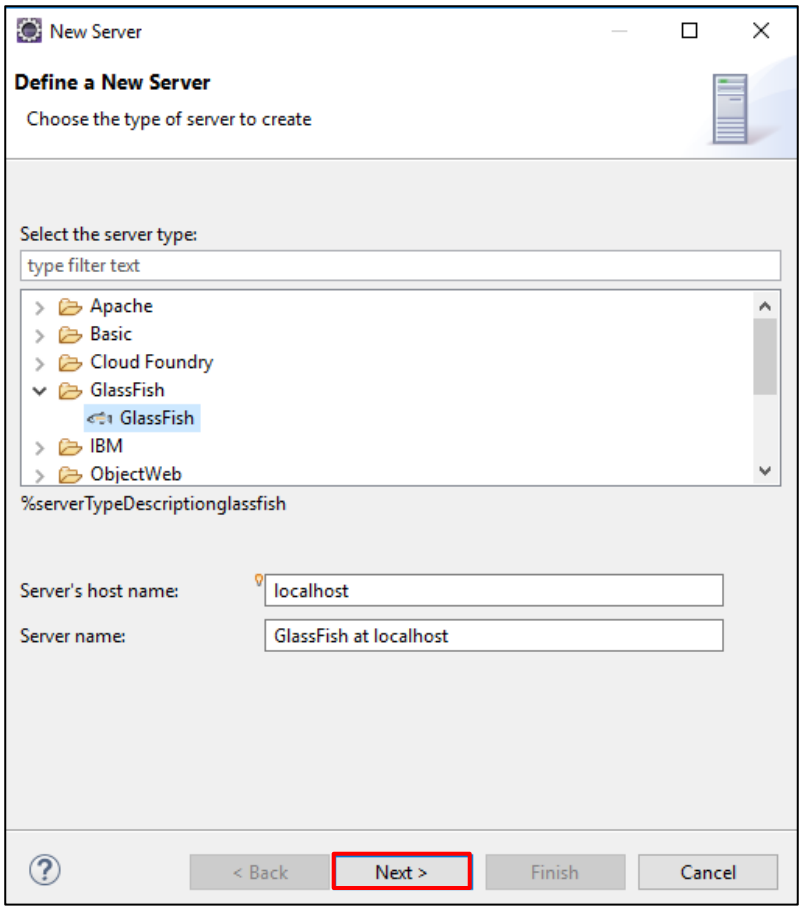


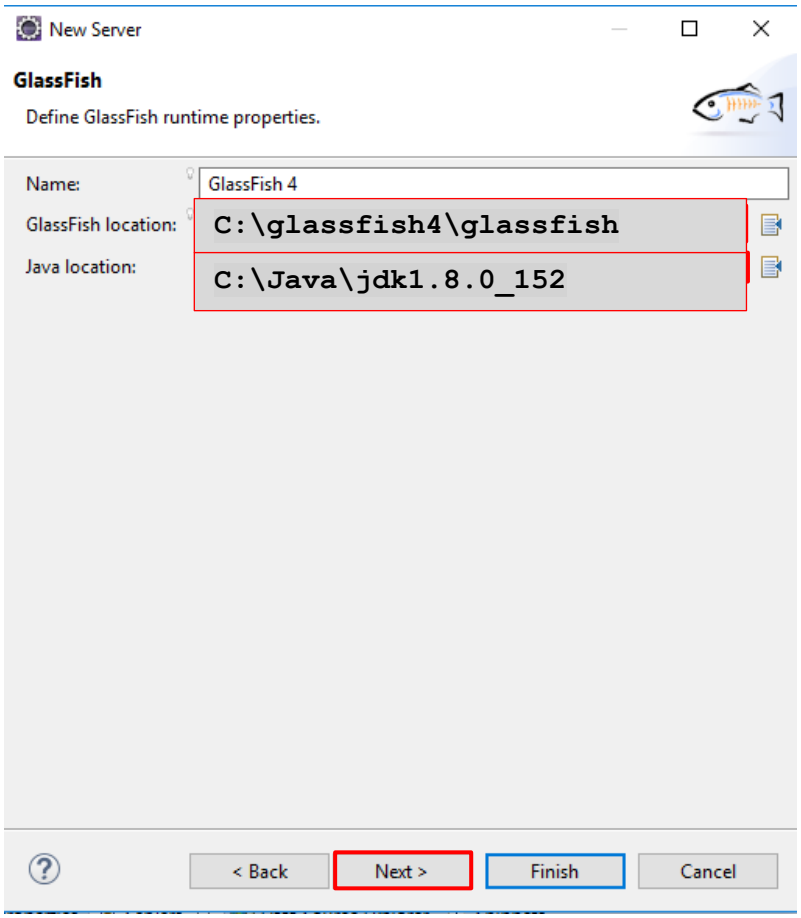


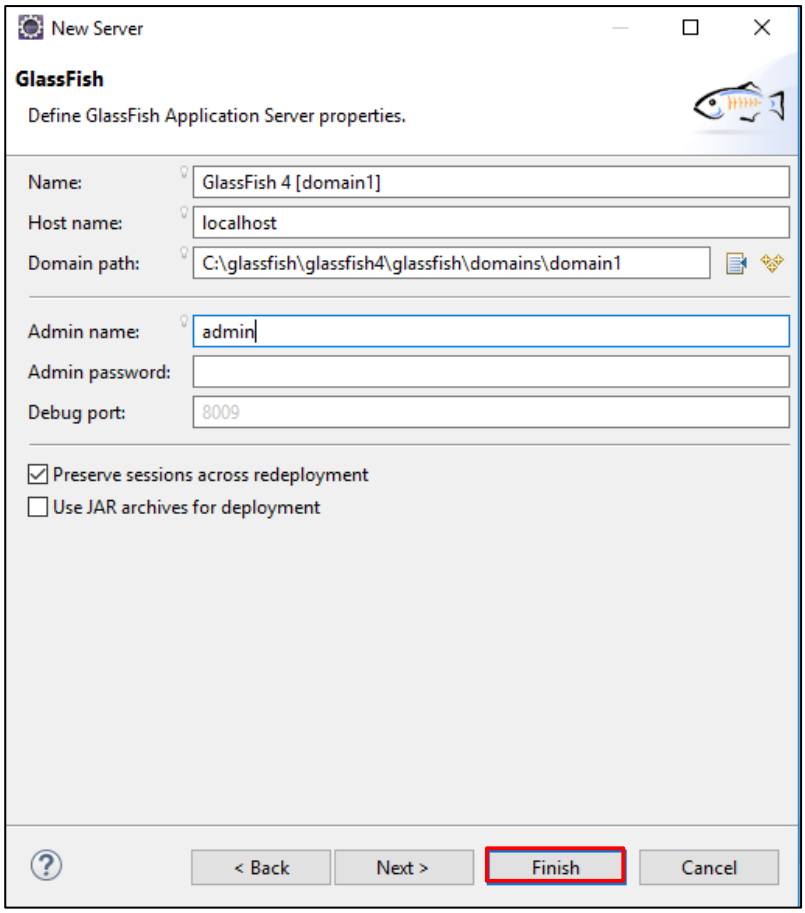
Then click on the Servers-tab which should be at the bottom area. Afterwards click on the blue underlined text to setup a new one.



Falls kein Glassfish Eintrag gefunden wird, diesen Schritt zuerst durchführen and Reboot.

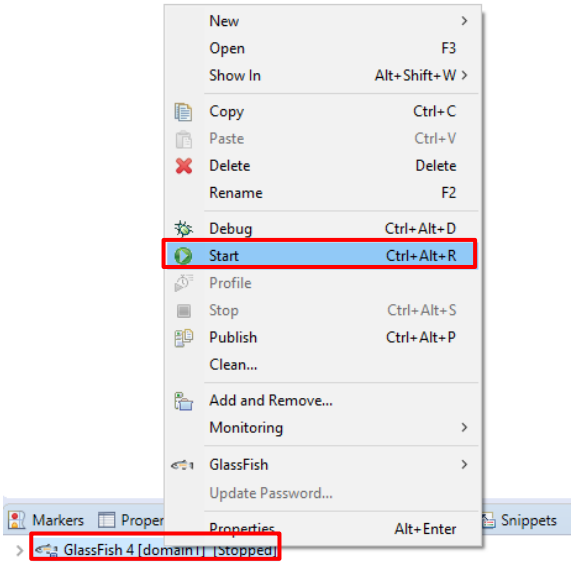

New Server
Define a New Server
Choose the type of server to create
Select the server type:
type filter text
Apache
Basic
Cloud Foundry
GlassFish
GlassFish
IBM
ObjectWeb
%serverTypeDescriptionglassfish
Server's host name:
localhost
Server name:
GlassFish at localhost
< Back
Next >
Finish
Cancel


New Server
GlassFish
Define GlassFish runtime properties.
Name:
GlassFish 4
GlassFish location:
C:\glassfish4\glassfish
Java location:
C:\Java\jdk1.8.0_152
< Back
Next >
Finish
Cancel

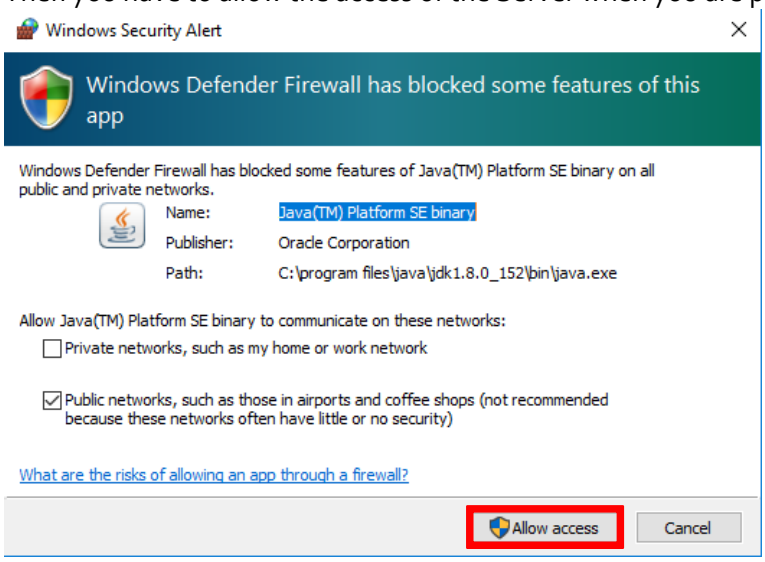

New Server
GlassFish
Define GlassFish Application Server properties.
Name:
GlassFish 4 [domain1]
Host name:
localhost
Domain path:
C:\glassfish\glassfish4\glassfish\domains\domain1
Admin name:
admin
Admin password:
Debug port:
8009
Preserve sessions across redeployment
Use JAR archives for deployment
< Back
Next >
Finish
Cancel

Start Glassfish Admin Console

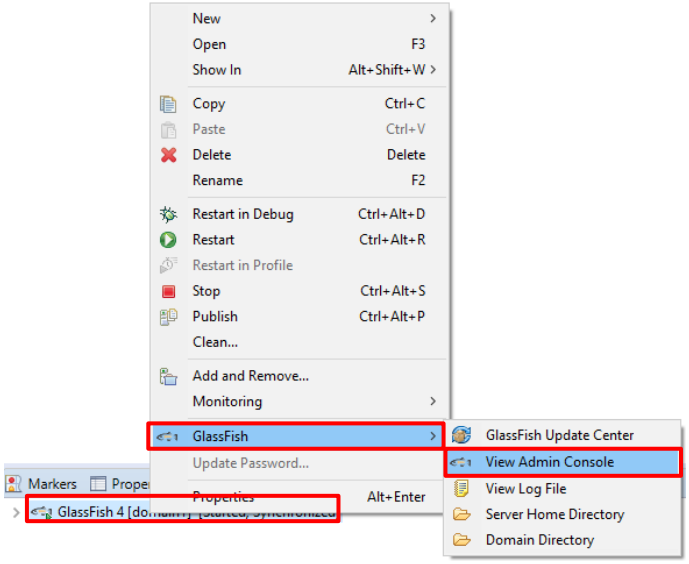
Right click the server and start it.



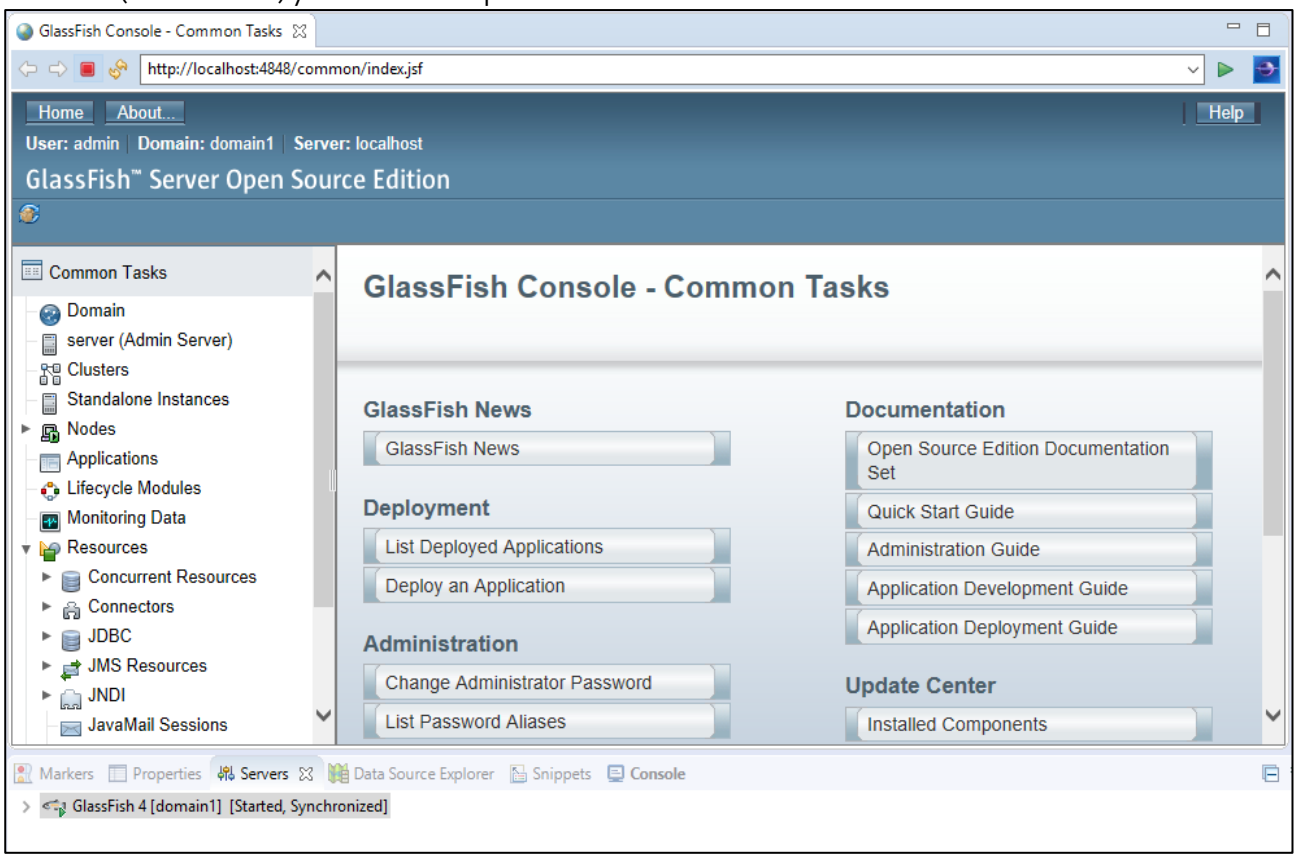
Then you have to allow the access of the Server when you are prompted.



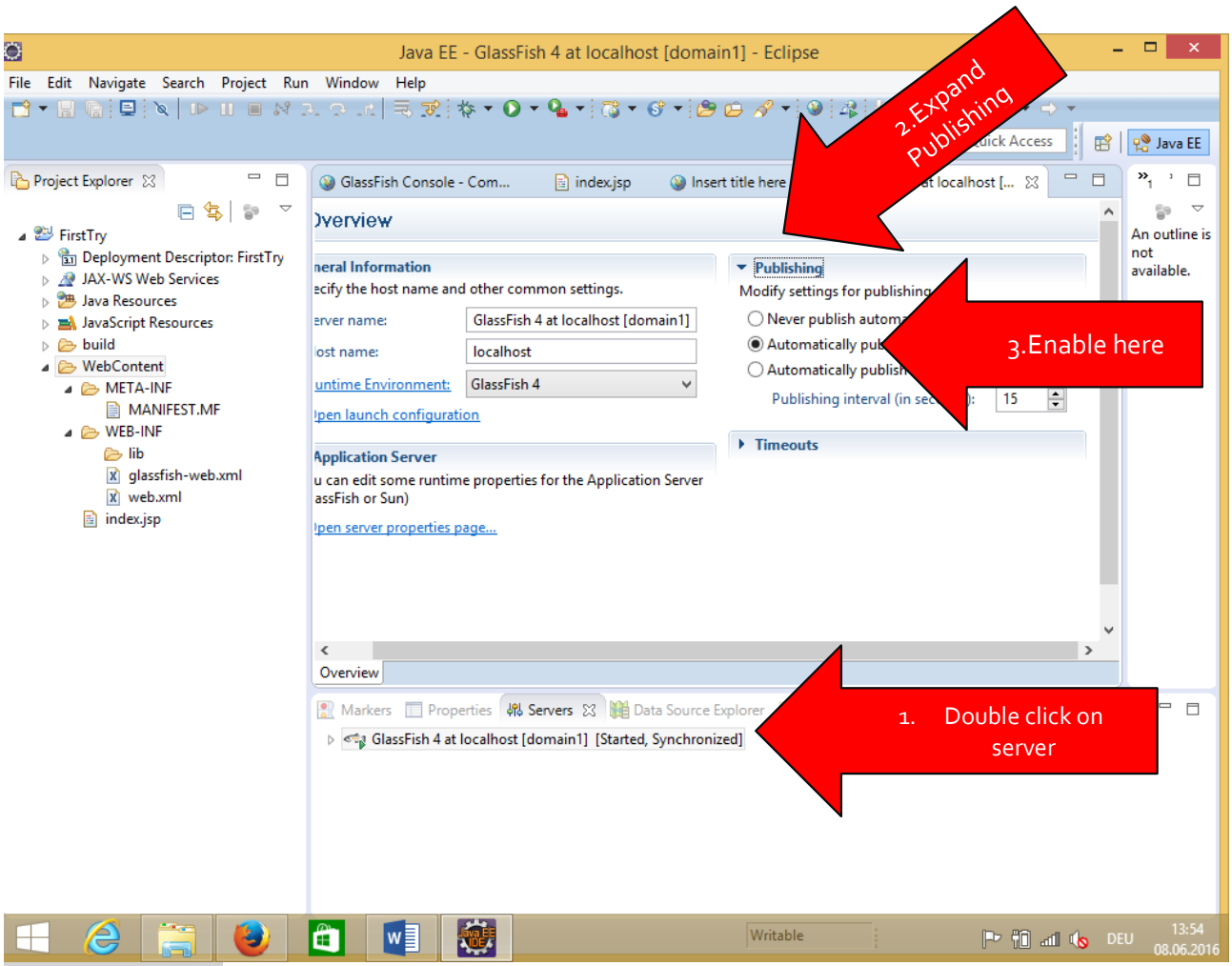
After that **right click** on Glassfish Select Glassfish -> View Admin Console



And then (after a while) you will see the picture down below.



Deploy whenever you press CTRL-S to your source file



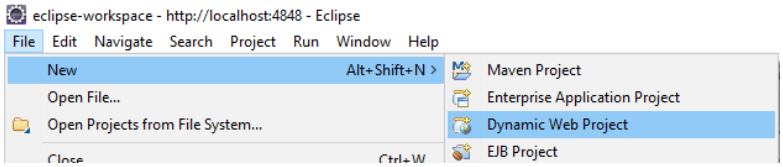
Ctrl-S (Save) to save the changes!

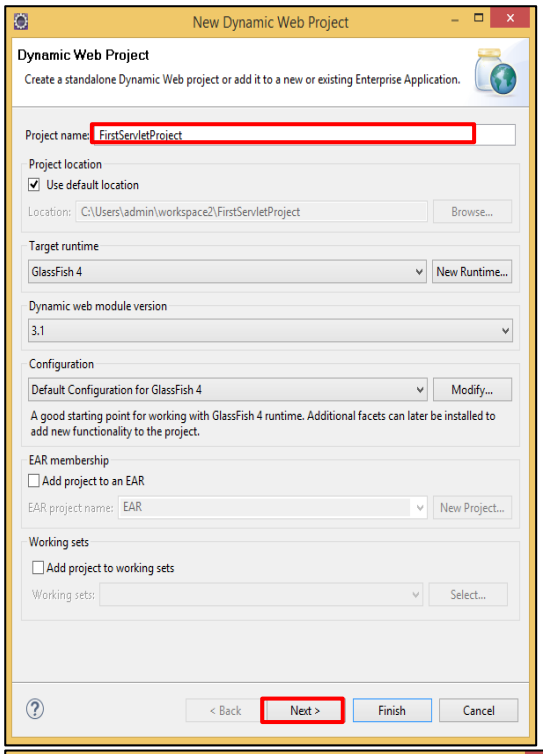
Testing JEE installation

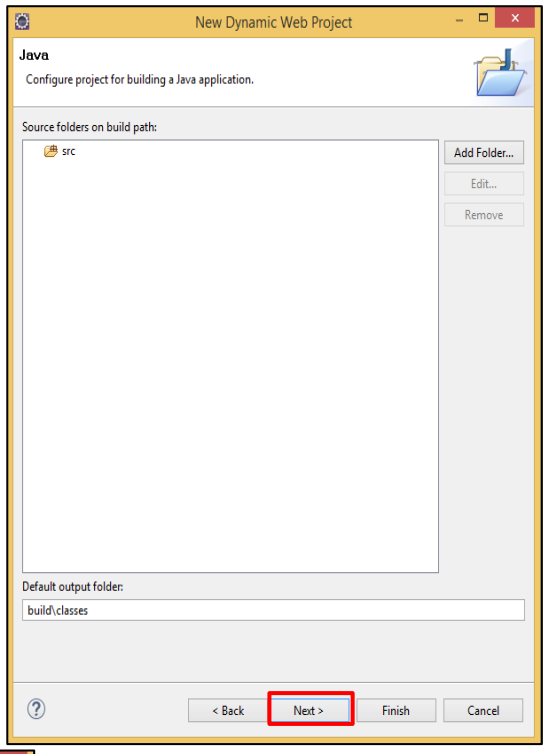
Test the Installation: Create first Servlet and JSP, deploy and run it

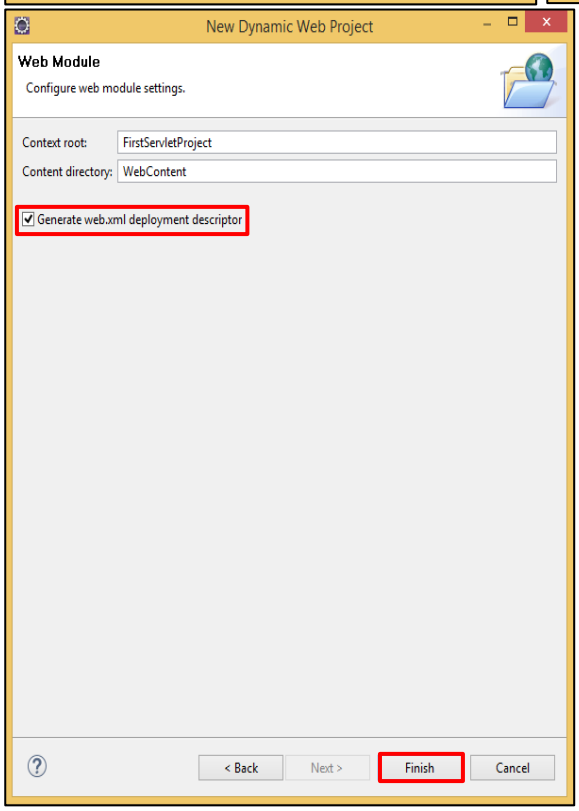
Right click in Project section of eclipse:

```
File -> New -> Dynamic Web Project
```

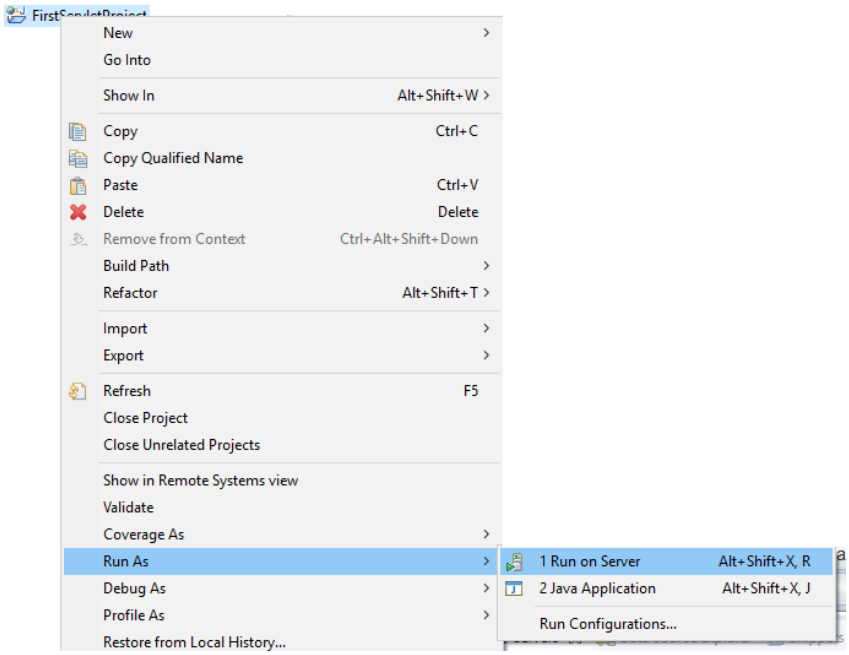


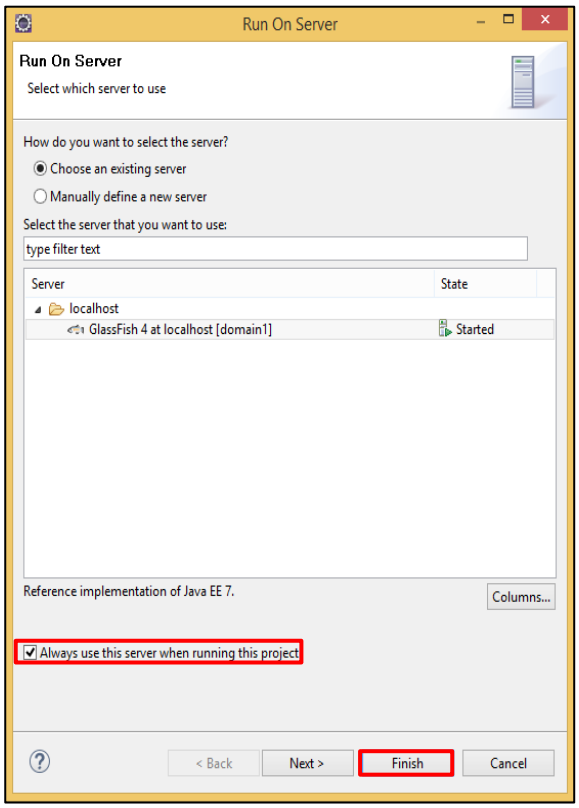




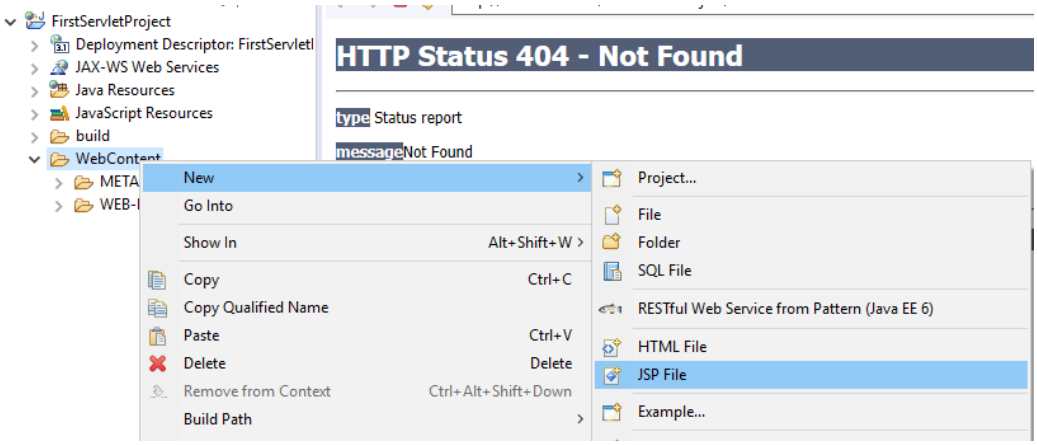


Right click the newly created `Project -> Run As -> run on server`





An **index.jsp** has been created in the WebContent folder of the project. Double-Click on **index.jsp** to open it in editor.
If you can't find the **index.jsp** file, then **right click** on:
WebContent-> New -> JSP File. (if you can't find JSP File then go: New -> Other -> Web -> JSP File) and change the filename to **index.jsp**. Now open it.



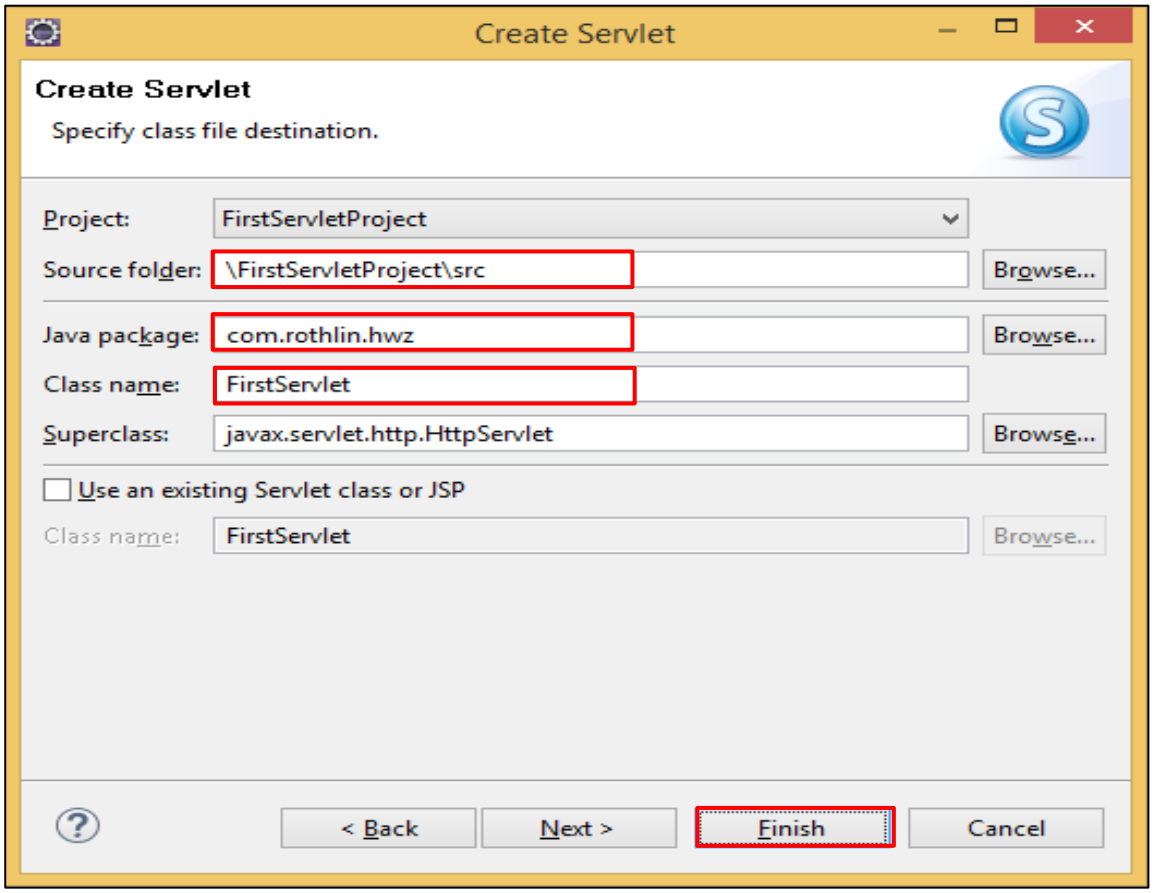
When you go to your **index.jsp** opened in the editor, replace its code with the code below and run it.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>GlassFish JSP Page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

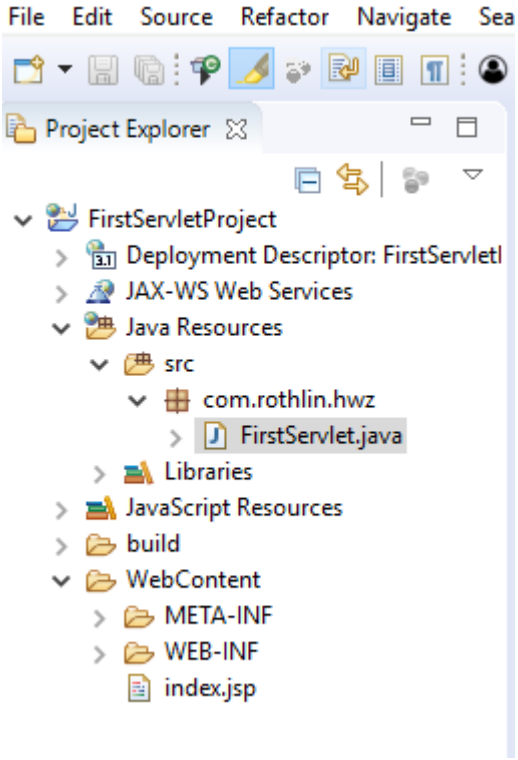
You should now see a big Hello World! If not, you probably did something wrong in the previous steps or you need to fully reload the page. Just press CTRL+F5 and give it a bit of time.

Hint: Just press CTRL-S (to save the changes in the source code) and eclipse will automatically deploy the changes to the application-server.

File -> New -> Other -> Web -> Servlet



Your project structure should look like this:



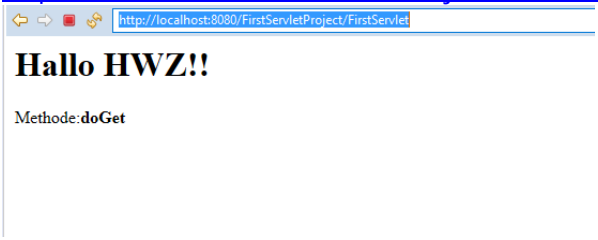
First Servlet Code

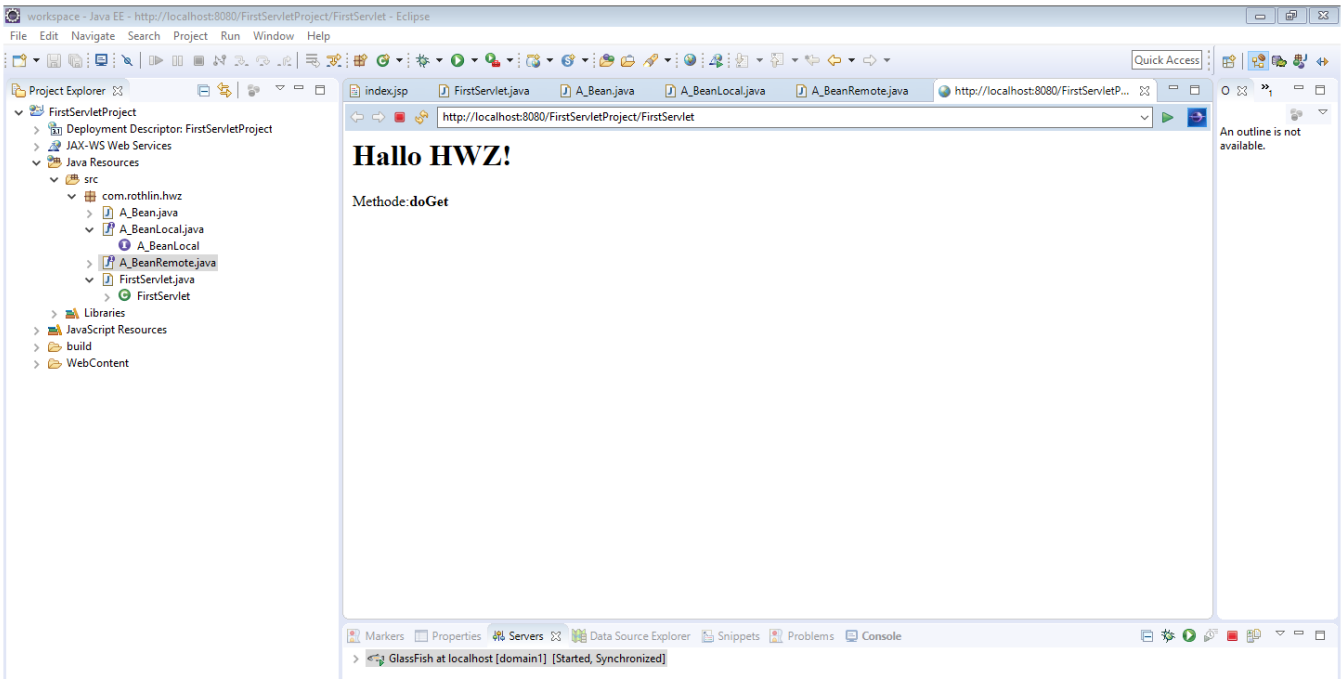
Replace in the generated Servlet-Class all methods with the code from below, press Crtl+Shift+O and save it.

```
package com.rothlin.hwz;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;

        public FirstServlet() {
                super();
                System.out.println("FirstServlet ctr is called! Object created!!!");
        }

        public void init(ServletConfig servletConfig) throws ServletException {
                System.out.println("FirstServlet init() is called!");
        }

        public void destroy() {
                System.out.println("FirstServlet destroy() is called!");
        }

        private void serveRequest(HttpServletRequest request, HttpServletResponse response, String methode)
                throws ServletException, IOException {
                response.setContentType("text/html");
                StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
                sb.append("<H1>Hallo HWZ!</H1> Methode:<B>" + methode + "</B><br/>\n");
                sb.append("</body>\n</html>\n");
                response.getWriter().print(sb);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
                serveRequest(request, response, "doGet");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
                serveRequest(request, response, "doPost");
        }
}
```

Testing FirstServlet.java

Run the project on server…
http://localhost:8080/FirstServletProject/FirstServlet should show you this:

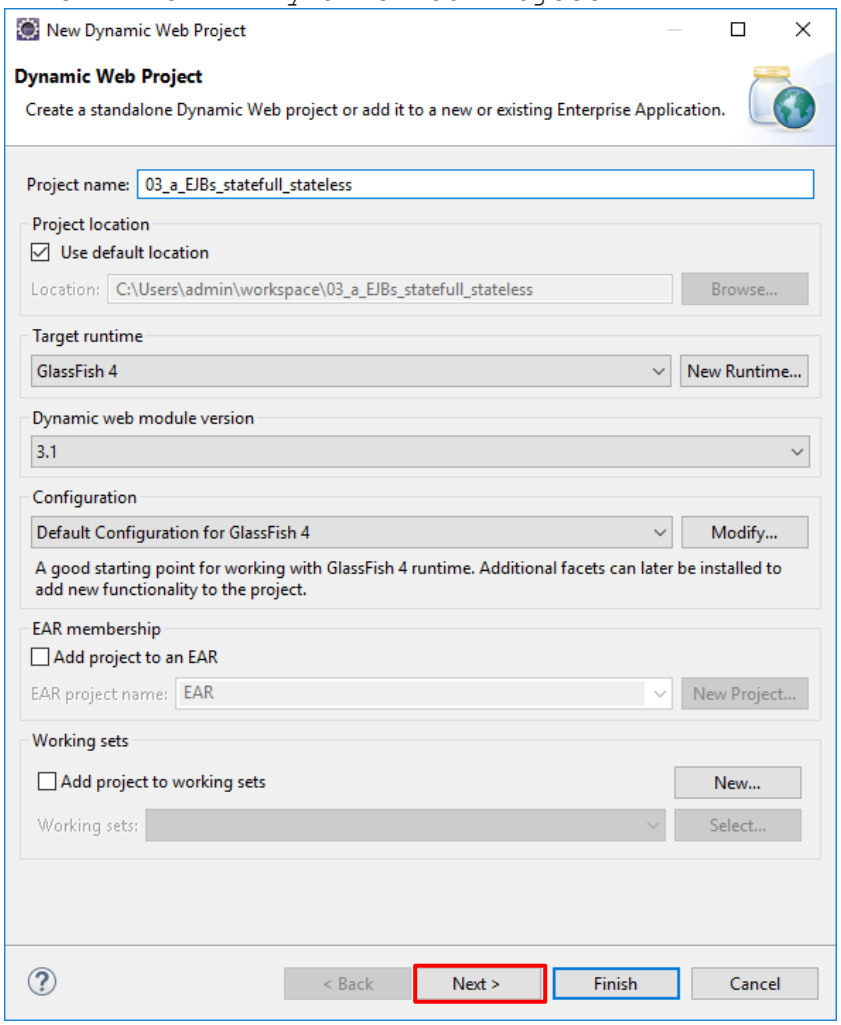


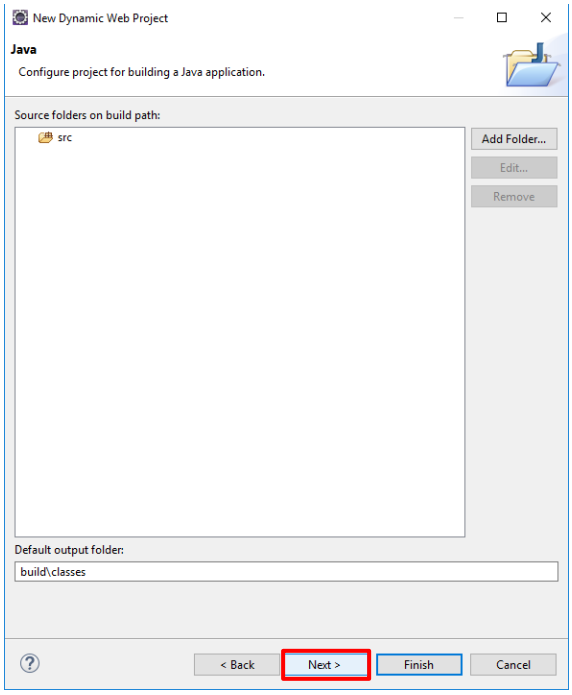


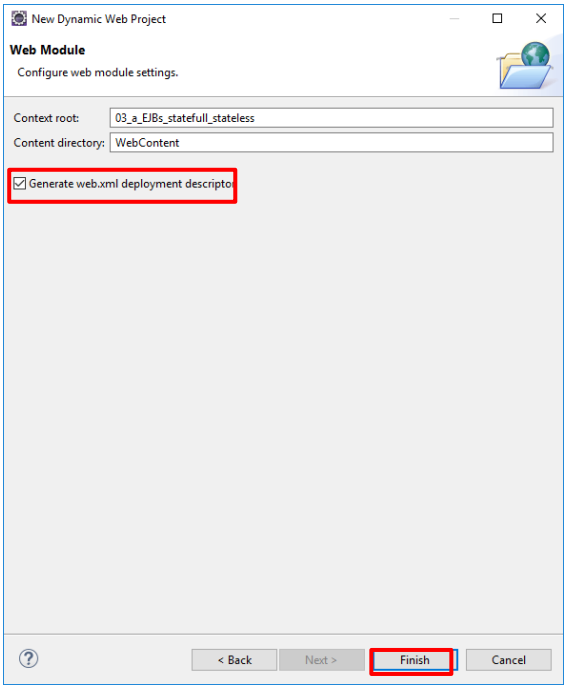
JEE development

EJB (stateless and stateful) gets called from Servlet / JSP (in same Web-App)

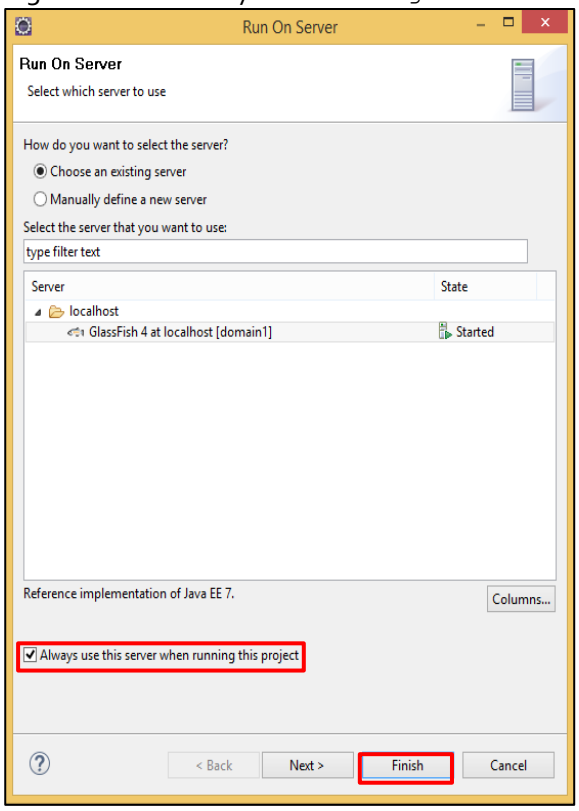
```
File -> New -> Dynamic Web Project
```





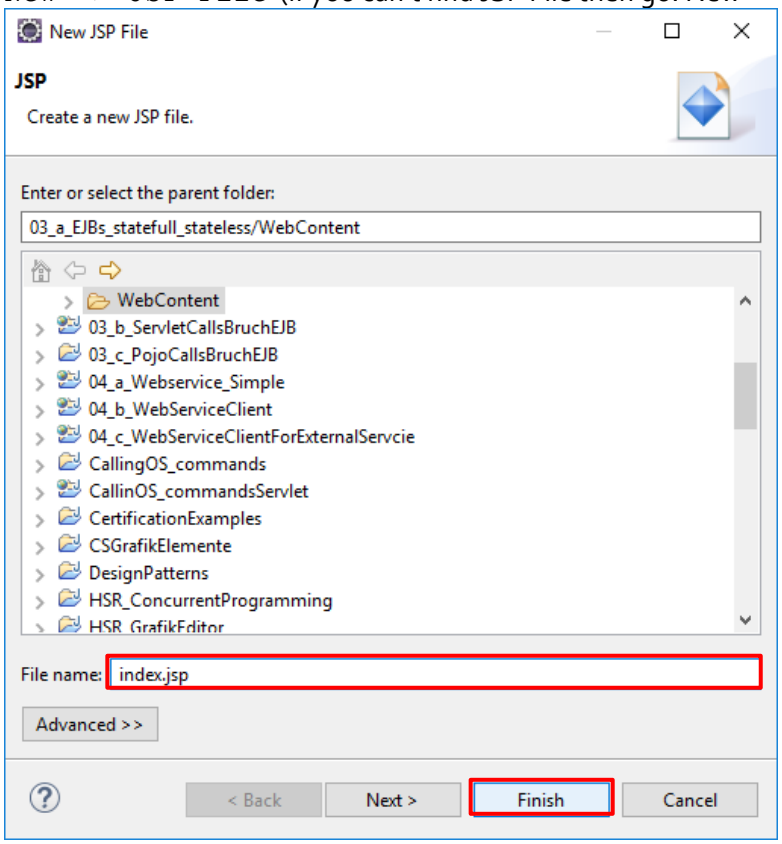


Right click the newly created `Project -> Run As -> Run on Server`



Create a JSP to test project set-up:

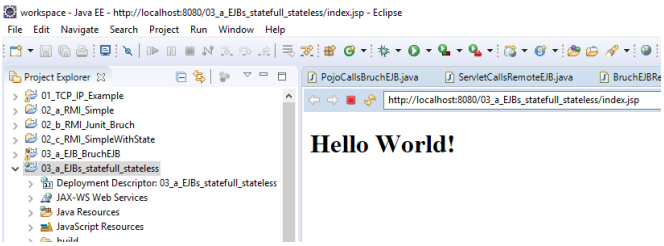
Right click on project:
`New -> JSP File` (if you can't find JSP File then go: New -> Other -> Web -> JSP File)



When you go your **index.jsp** opened in the editor, replace its code with the code below and run it.
You should now see a big Hello World! If not, you probably did something wrong in the previous steps.

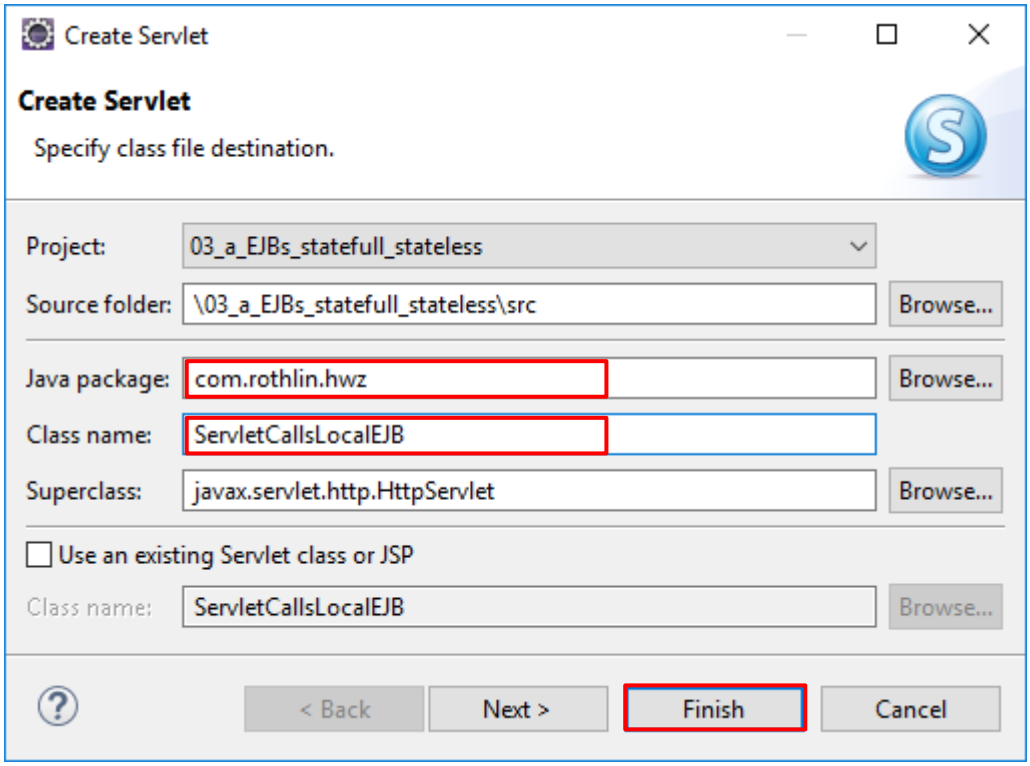
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>GlassFish JSP Page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Save and run it:



Right click on project:

`New -> Servlet`



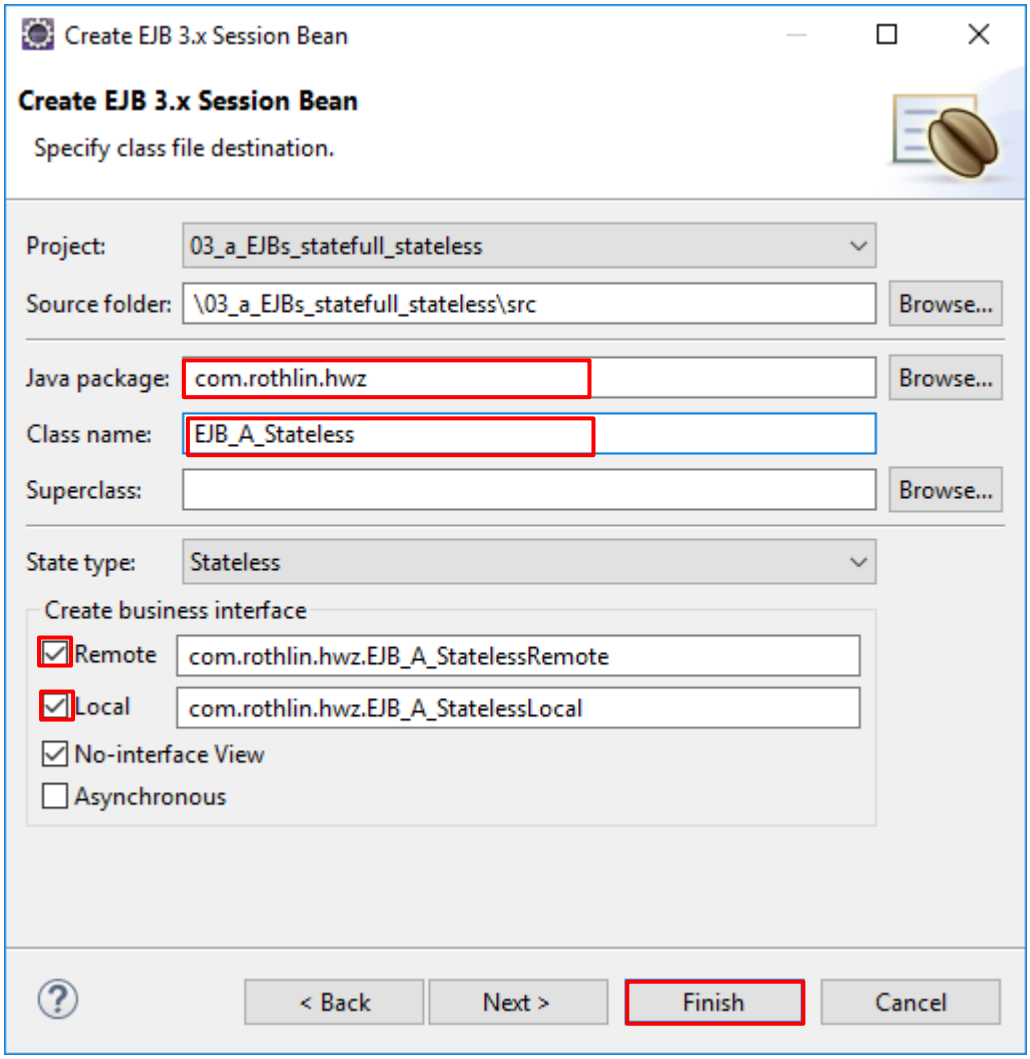
Replace in the generated Servlet-Class all methods with this code:

```
// ServletCallsLocalEJB belongs to 03_a_EJBs_statefull_stateless
// --------------------            -----------------------------
package com.rothlin.hwz;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ServletCallsLocalEJB")
public class ServletCallsLocalEJB extends HttpServlet {
	private static final long serialVersionUID = 1L;

	public ServletCallsLocalEJB() {
		super();
		System.out.println("ServletCallsLocalEJB::ctr() is called! Object created!!!");
	}

	public void init(ServletConfig servletConfig) throws ServletException {
		System.out.println("ServletCallsLocalEJB::init() is called!");
	}

	public void destroy() {
		System.out.println("ServletCallsLocalEJ::destroy() is called!");
	}

	private void serveRequest(HttpServletRequest request, HttpServletResponse response, String methode)
			throws ServletException, IOException {
		response.setContentType("text/html");
		StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
		sb.append("<H1>Hallo from ServletCallsLocalEJB!</H1> Methode:<B>" + methode + "</B><br/>\n");
		sb.append("</body>\n</html>\n");
		response.getWriter().print(sb);
		}

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		serveRequest(request, response, "doGet");
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		serveRequest(request, response, "doPost");
	}
}
```

Save and run it:

Create **stateless** EJB in the same project (Web-App):

Right click on project:
`New -> Other -> EJB -> Session Bean (EJB 3.x)`



Code for **stateless** EJB:

EJB_A_StatelessRemote.java

```
// EJB_A_StatelessRemote belongs to 03_a_EJBs_statefull_stateless
// ---------------------           ----------------------------
package com.rothlin.hwz;

import javax.ejb.Remote;

@Remote
public interface EJB_A_StatelessRemote {
        public String pingRemote();
}
```

EJB_A_StatelessLocal.java

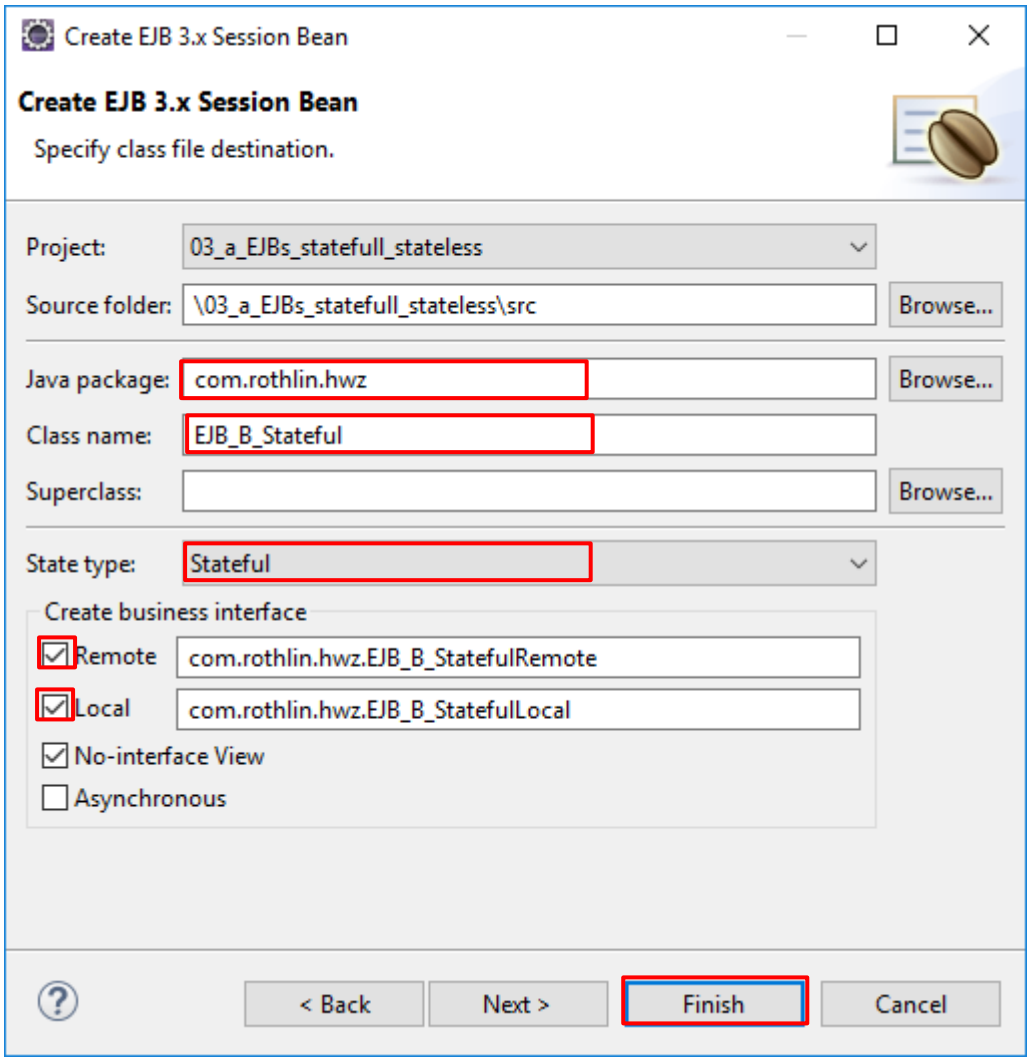
```
// EJB_A_StatelessLocal belongs to 03_a_EJBs_statefull_stateless
// --------------------            ----------------------------
package com.rothlin.hwz;

import javax.ejb.Local;

@Local
public interface EJB_A_StatelessLocal {
        public String pingLocal();
        public String pingRemote();
}
```

EJB_A_Stateless.java

```
// EJB_A_Stateless belongs to 03_a_EJBs_statefull_stateless
// ---------------            ----------------------------
package com.rothlin.hwz;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.LocalBean;
import javax.ejb.Stateless;

@Stateless
@LocalBean
public class EJB_A_Stateless implements EJB_A_StatelessRemote, EJB_A_StatelessLocal {
        private int counter;

    public EJB_A_Stateless() {
        System.out.println("EJB_A_Stateless::ctr() called!");
    }

    @PostConstruct
    public void startup() {
        System.out.println("EJB_A_Stateless::startup() called!");
        counter = 10;
    }

    @PreDestroy
    public void cleanup() {
        System.out.println("EJB_A_Stateless::cleanup() called!");
    }


        @Override
        public String pingRemote() {
        counter++;
        return "EJB_A_Stateless::pingRemoteOnly(): counter: " + counter;
        }

        @Override
        public String pingLocal() {
        counter++;
        return "EJB_A_Stateless::pingLocal(): counter: " + counter;
        }
}
```

Create **stateful** EJB in the same project (Web-App):

Right click on project:
`New -> Other -> EJB -> Session Bean (EJB 3.x)`



Code for **stateful** EJB:

EJB_B_StatefulRemote.java

```
// EJB_B_StatefulRemote belongs to 03_a_EJBs_statefull_stateless
// --------------------           ----------------------------
package com.rothlin.hwz;

import javax.ejb.Remote;

@Remote
public interface EJB_B_StatefulRemote {
        public String pingRemote();
}
```

EJB_B_StatefulLocal.java

```
// EJB_B_StatefulLocal belongs to 03_a_EJBs_statefull_stateless
// -------------------            ----------------------------
package com.rothlin.hwz;

import javax.ejb.Local;

@Local
public interface EJB_B_StatefulLocal {
        public String pingLocal();
        public String pingRemote();
}
```

EJB_B_Stateful.java

```
// EJB_B_Stateful belongs to 03_a_EJBs_statefull_stateless
// --------------            ----------------------------
package com.rothlin.hwz;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.LocalBean;
import javax.ejb.Stateful;

@Stateful
@LocalBean
public class EJB_B_Stateful implements EJB_B_StatefulRemote, EJB_B_StatefulLocal {
        private int counter;

    public EJB_B_Stateful() {
        System.out.println("EJB_B_Stateful::ctr() called!");
    }

    @PostConstruct
    public void startup() {
        System.out.println("EJB_B_Stateful::startup() called!");
        counter = 10;
    }

    @PreDestroy
    public void cleanup() {
        System.out.println("EJB_B_Stateful::cleanup() called!");
    }


        @Override
        public String pingRemote() {
        counter++;
        return "EJB_B_Stateful::pingRemoteOnly(): counter: " + counter;
        }

        @Override
        public String pingLocal() {
        counter++;
        return "EJB_B_Stateful::pingLocal(): counter: " + counter;
        }
}
```

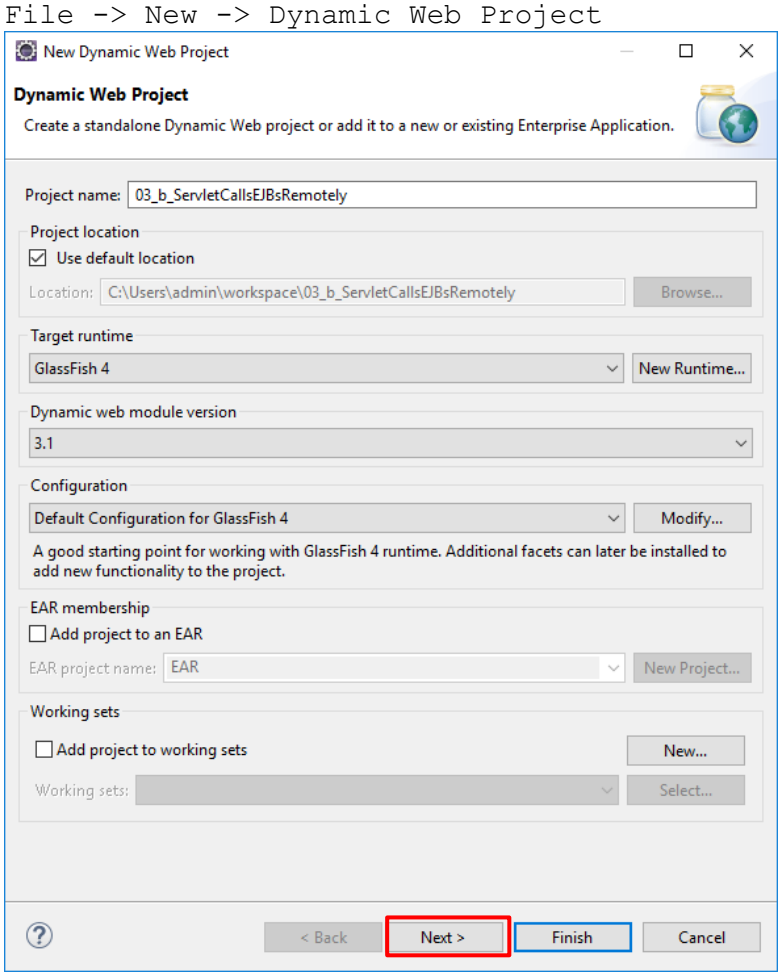
Extend your Servlet-Code with the yellow lines:

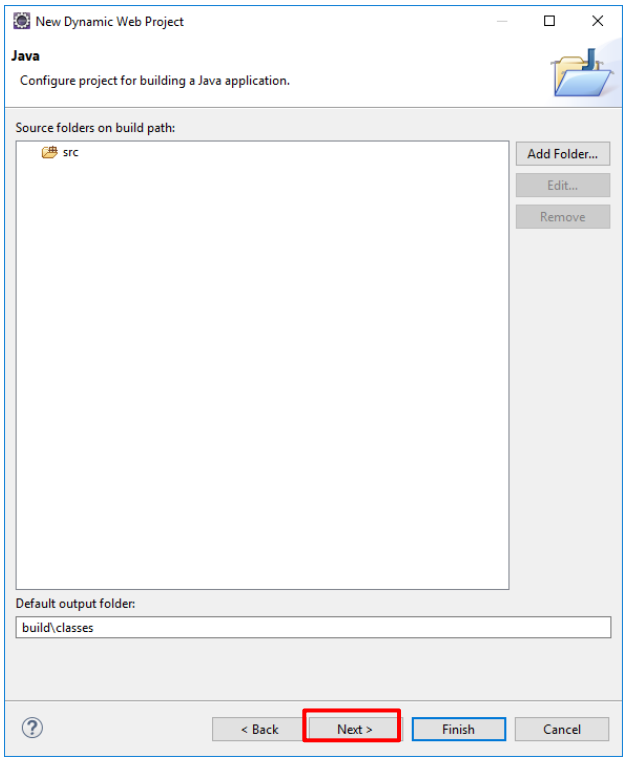
```
// ServletCallsLocalEJB belongs to 03_a_EJBs_statefull_stateless
// --------------------            ----------------------------
package com.rothlin.hwz;

import javax.ejb.EJB;
import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ServletCallsLocalEJB")
public class ServletCallsLocalEJB extends HttpServlet {
        private static final long serialVersionUID = 1L;

        @EJB
        EJB_A_Stateless aLocalEJB;

        @EJB
        EJB_B_Stateful bLocalEJB;

        public ServletCallsLocalEJB() {
                super();
                System.out.println("ServletCallsLocalEJB::ctr() is called! Object created!!!");
        }

        public void init(ServletConfig servletConfig) throws ServletException {
                System.out.println("ServletCallsLocalEJB::init() is called!");
        }

        public void destroy() {
                System.out.println("ServletCallsLocalEJ::destroy() is called!");
        }

        private void serveRequest(HttpServletRequest request, HttpServletResponse response, String methode)
                        throws ServletException, IOException {
                response.setContentType("text/html");
                StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
                sb.append("<H1>Hallo from ServletCallsLocalEJB!</H1> Methode:<B>" + methode +
"</B><br/>\n");
                sb.append("<BR/>" + aLocalEJB.pingLocal()  + "\n");
                sb.append("<BR/>" + aLocalEJB.pingRemote() + "<BR/>\n");

                sb.append("<BR/>" + bLocalEJB.pingLocal()  + "\n");
                sb.append("<BR/>" + bLocalEJB.pingRemote() + "\n");
                sb.append("</body>\n</html>\n");
                response.getWriter().print(sb);
                }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doGet");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doPost");
        }
}
```

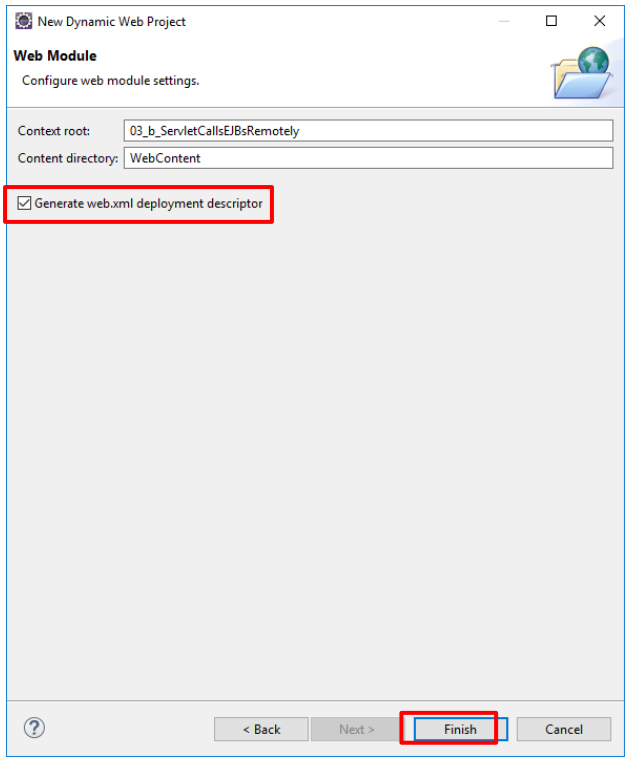
Save and run it. After a while you will see the following:

EJB (stateless and stateful) gets called from **Servlet** remotely

```
File -> New -> Dynamic Web Project
```

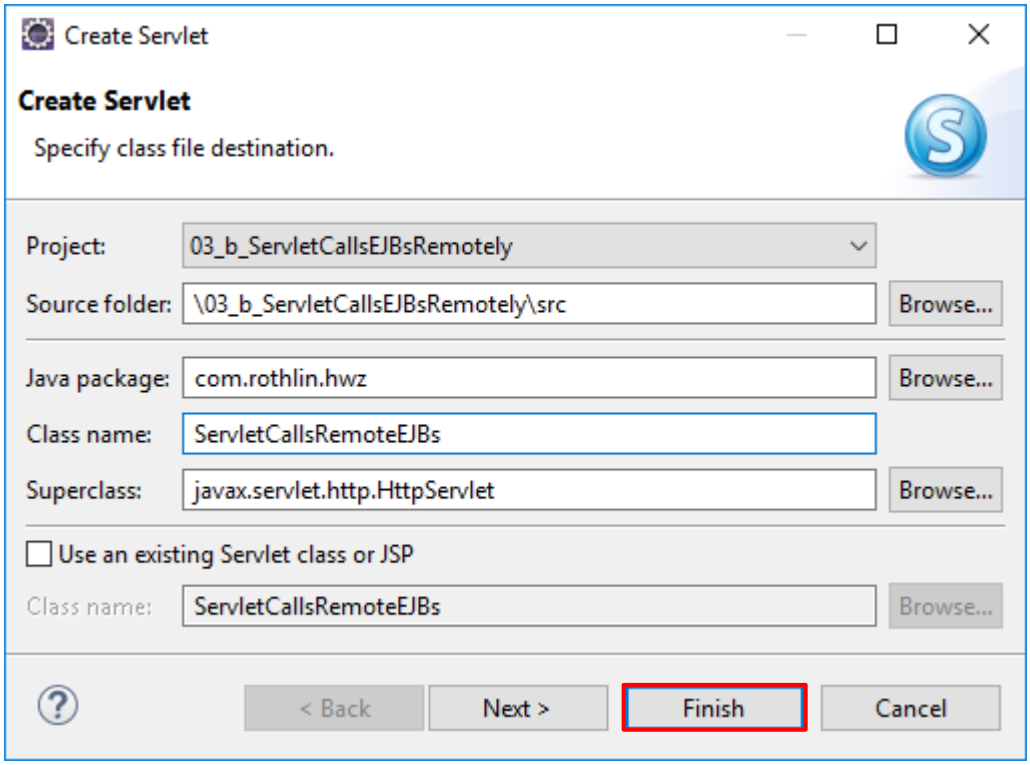






Right click on project:

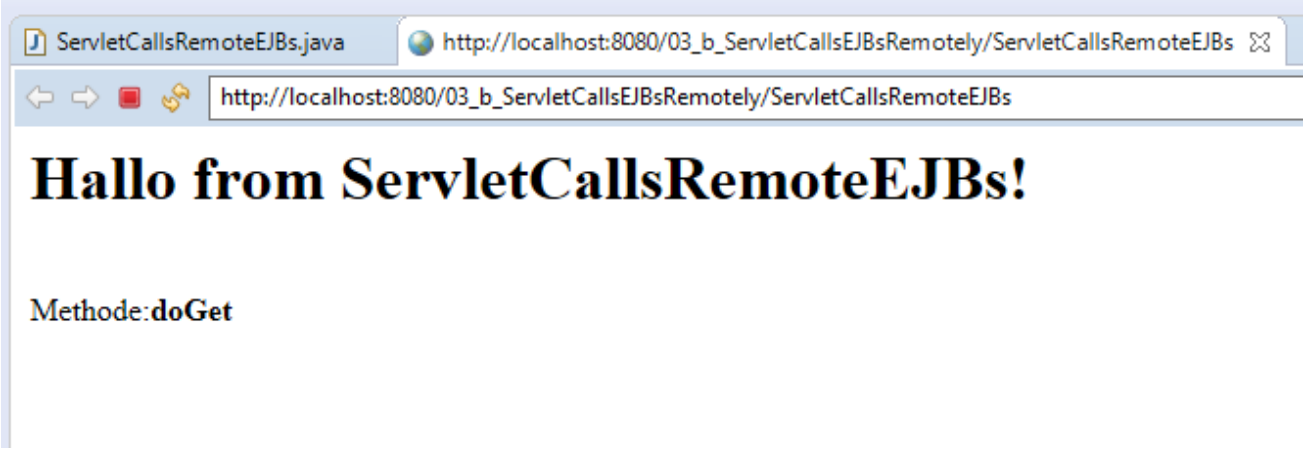
```
New -> Servlet
```



Replace in the generated Servlet-Class all methods with this code:

```
// ServletCallsRemoteEJBs belongs to 03_b_ServletCallsEJBsRemotely
// ----------------------            ----------------------------

package com.rothlin.hwz;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/ServletCallsRemoteEJBs")
public class ServletCallsRemoteEJBs extends HttpServlet {
        private static final long serialVersionUID = 1L;

        public ServletCallsRemoteEJBs() {
                System.out.println("ServletCallsRemoteEJBs::ctr() is called! Object created!!!");
        }

        public void init(ServletConfig servletConfig) throws ServletException {
                System.out.println("ServletCallsRemoteEJBs::init() is called!");
        }

        public void destroy() {
                System.out.println("ServletCallsRemoteEJBs::destroy() is called!");
        }

        private void serveRequest(HttpServletRequest request, HttpServletResponse response, String methode)
                        throws ServletException, IOException {
                response.setContentType("text/html");
                StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
                sb.append("<H1>Hallo from ServletCallsRemoteEJBs!</H1><BR/>\n");
                sb.append("Methode:<B>" + methode + "</B><br/>\n");
                sb.append("</body>\n</html>\n");
                response.getWriter().print(sb);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doGet");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doPost");
        }
}
```

Save and run it.



Extend your Servlet-Code with the yellow lines:

```
// ServletCallsRemoteEJBs belongs to 03_b_ServletCallsEJBsRemotely
// ----------------------            ----------------------------
package com.rothlin.hwz;

import java.io.IOException;
import javax.naming.InitialContext;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ServletCallsRemoteEJBs")
public class ServletCallsRemoteEJBs extends HttpServlet {
        private static final long serialVersionUID = 1L;

        private static final String jndiHost                        = "127.0.0.1";
        private static final String jndiPort                        = "3700";
        private static final String jndiName__EJB_A_StatelessRemote =
"java:global/03_a_EJBs_statefull_stateless/EJB_A_Stateless!com.rothlin.hwz.EJB_A_StatelessRemote";
        private static final String jndiName__EJB_B_StatefulRemote  =
"java:global/03_a_EJBs_statefull_stateless/EJB_B_Stateful!com.rothlin.hwz.EJB_B_StatefulRemote";

        private EJB_A_StatelessRemote ejbA = null;
        private EJB_B_StatefulRemote  ejbB = null;

        public ServletCallsRemoteEJBs() {
                System.out.println("ServletCallsRemoteEJBs::ctr() is called! Object created!!!");
        }

        public void init(ServletConfig servletConfig) throws ServletException {
                System.out.println("ServletCallsRemoteEJBs::init() is called!");
                System.setProperty("org.omg.CORBA.ORBInitialHost", jndiHost);
                try {
                        InitialContext ctx = new InitialContext();
                        ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", jndiHost);
                        ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", jndiPort);
                        ejbA = (EJB_A_StatelessRemote) ctx.lookup(jndiName__EJB_A_StatelessRemote);
                        ejbB = (EJB_B_StatefulRemote)  ctx.lookup(jndiName__EJB_B_StatefulRemote);
                }
                catch (Exception ex) {
                        System.out.println("Could not get remote-object bean!" + ex);
                }
        }

        public void destroy() {
                System.out.println("ServletCallsRemoteEJBs::destroy() is called!");
        }

        private void serveRequest(HttpServletRequest request, HttpServletResponse response, String methode)
                        throws ServletException, IOException {
                response.setContentType("text/html");
                StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
                sb.append("<H1>Hallo from ServletCallsRemoteEJBs!</H1><BR/>\n");
                sb.append("Methode:<B>" + methode + "</B><br/>\n");
                sb.append("ejbA.pingRemote():<B>" + ejbA.pingRemote() + "</B><br/>\n");
                sb.append("ejbB.pingRemote():<B>" + ejbB.pingRemote() + "</B><br/>\n");
                sb.append("</body>\n</html>\n");
                response.getWriter().print(sb);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doGet");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                serveRequest(request, response, "doPost");
        }
}
```

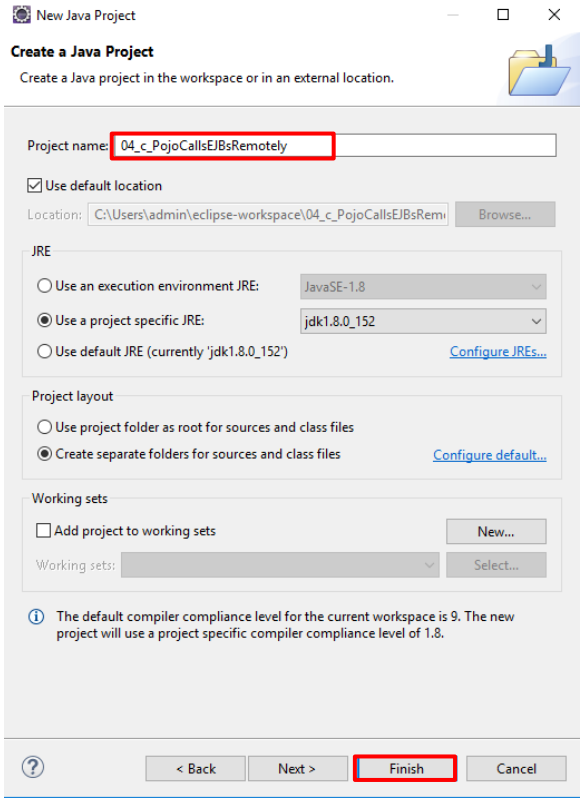
Copy the two Remote-Interface to your project (**`EJB_A_StatelessRemote, EJB_B_StatefulRemote`**)

Save and run it.

EJB (stateless and stateful) gets called from a **Java-Client (POJO)**

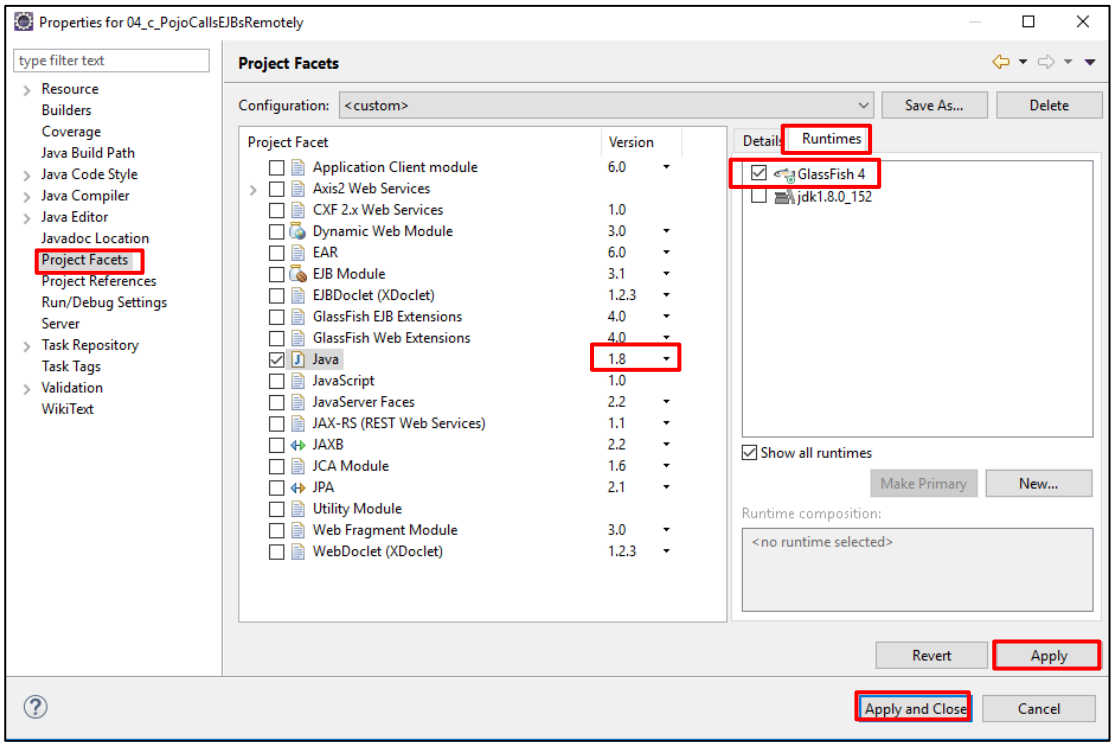
Right click on project:

`New -> Java Project`

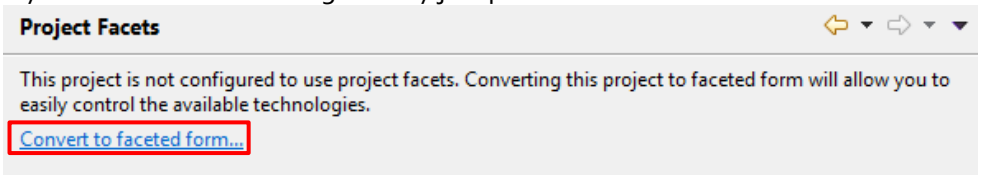


Set it up as an Glassfish-Client

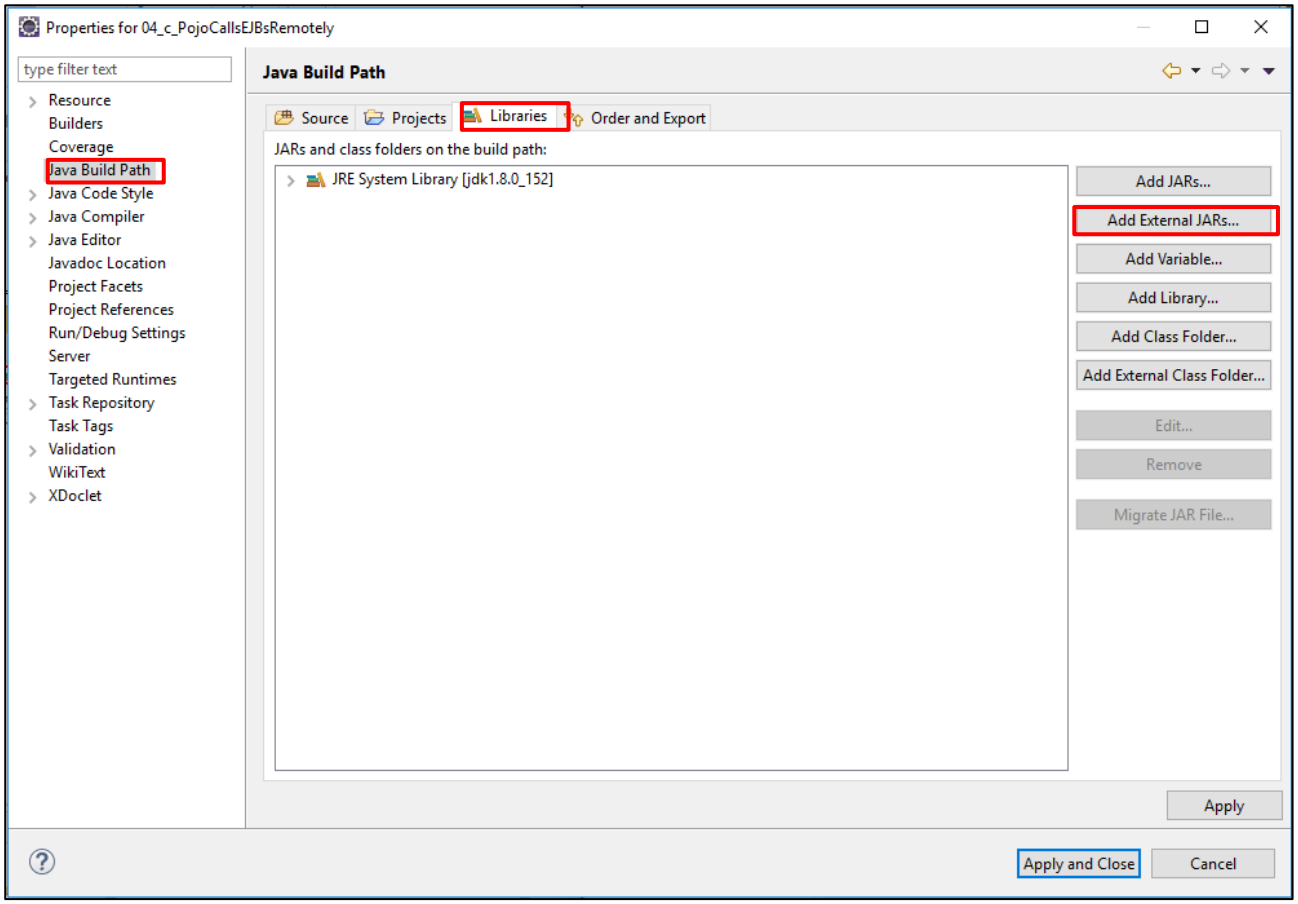
`Right click -> Properties -> Project Facets -> Runtimes -> Glassfish 4`

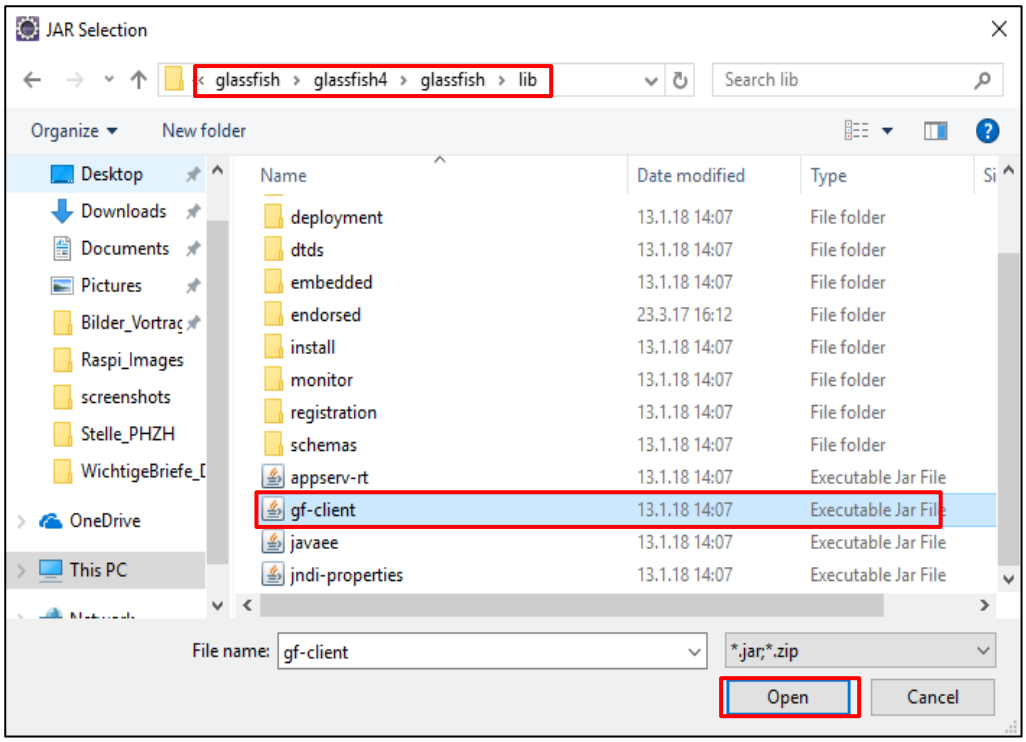


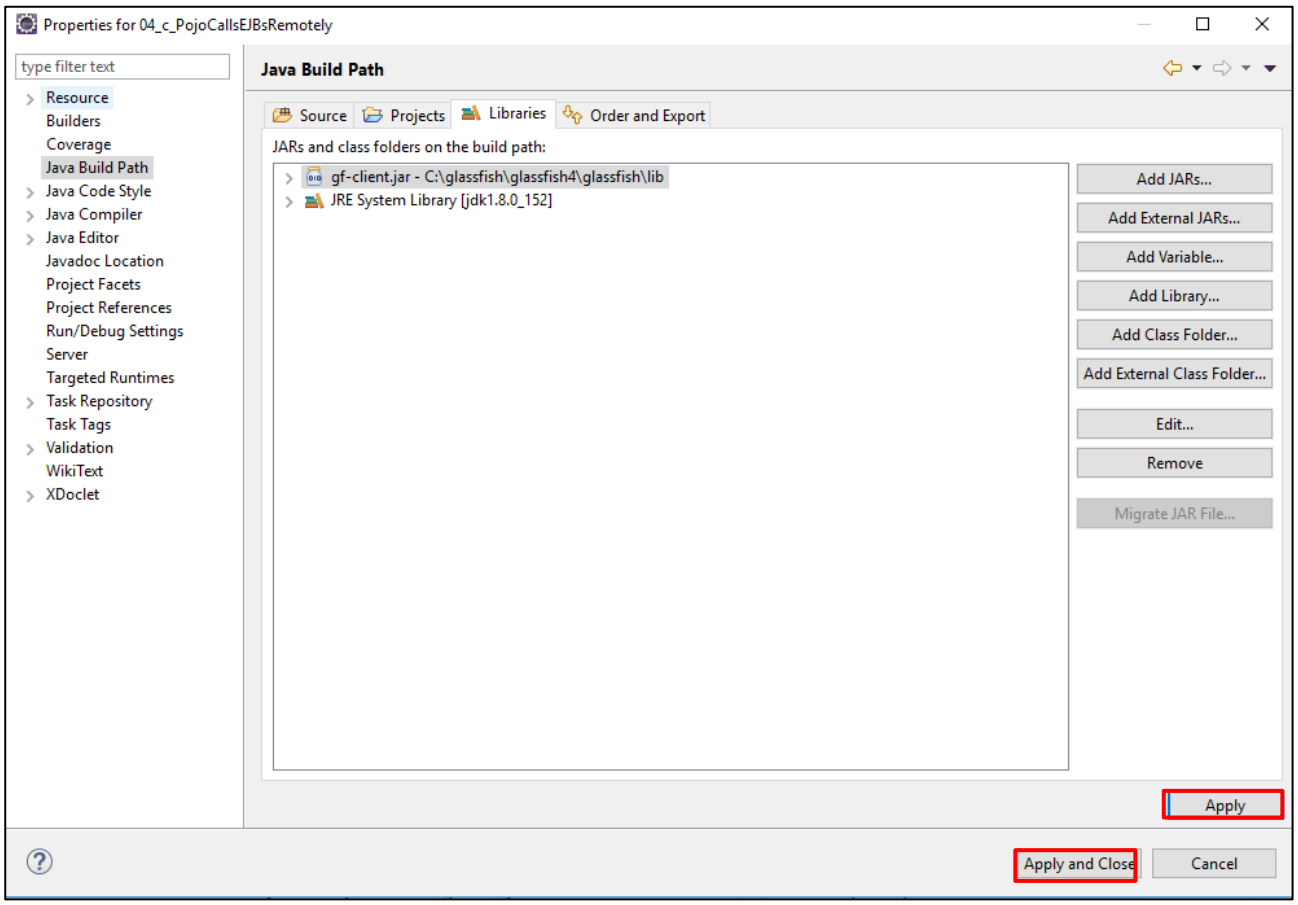
If you don't see this view right away just press this.



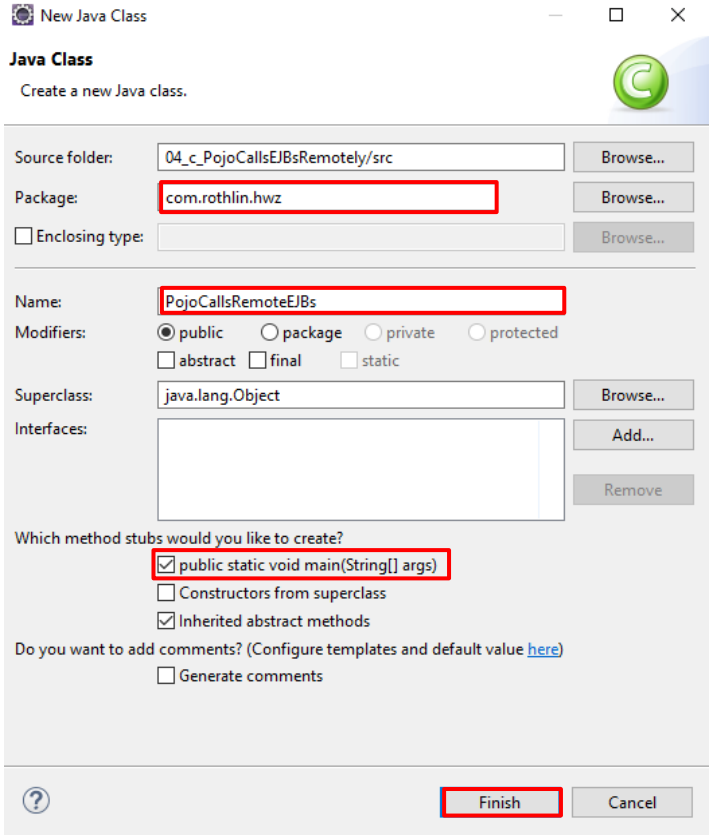
Right click -> Properties -> Java Build Path -> Libraries




JAR Selection
glassfish › glassfish4 › glassfish › lib
Search lib
Organize
New folder
Desktop
Downloads
Documents
Pictures
Bilder_Vortrag
Raspi_Images
screenshots
Stelle_PHZH
WichtigeBriefe_D
OneDrive
This PC
Name
Date modified
Type
deployment 13.1.18 14:07 File folder
dtds 13.1.18 14:07 File folder
embedded 13.1.18 14:07 File folder
endorsed 23.3.17 16:12 File folder
install 13.1.18 14:07 File folder
monitor 13.1.18 14:07 File folder
registration 13.1.18 14:07 File folder
schemas 13.1.18 14:07 File folder
appserv-rt 13.1.18 14:07 Executable Jar File
gf-client 13.1.18 14:07 Executable Jar File
javaee 13.1.18 14:07 Executable Jar File
jndi-properties 13.1.18 14:07 Executable Jar File
File name: gf-client
.jar;.zip
Open
Cancel


Properties for 04_c_PojoCallsEJBsRemotely
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project References
Run/Debug Settings
Server
Targeted Runtimes
Task Repository
Task Tags
Validation
WikiText
XDoclet
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
gf-client.jar - C:\glassfish\glassfish4\glassfish\lib
JRE System Library [jdk1.8.0_152]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel

`File -> New -> Other -> Java -> Class`

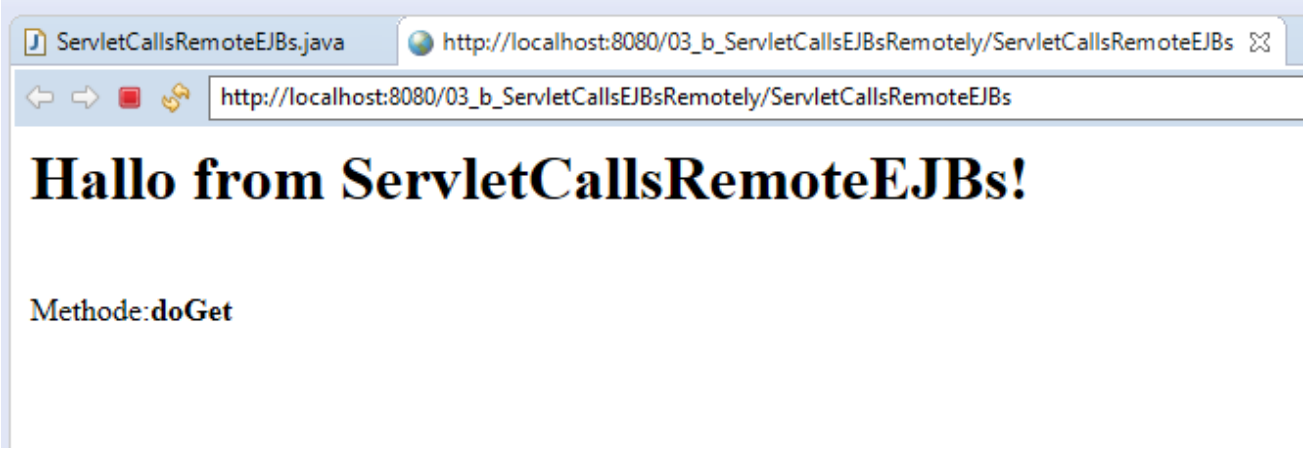


Copy the two Remote-Interface to your project (**`EJB_A_StatelessRemote`**, **`EJB_B_StatefulRemote`**)

Replace in code in the generated Java-Class with this code:

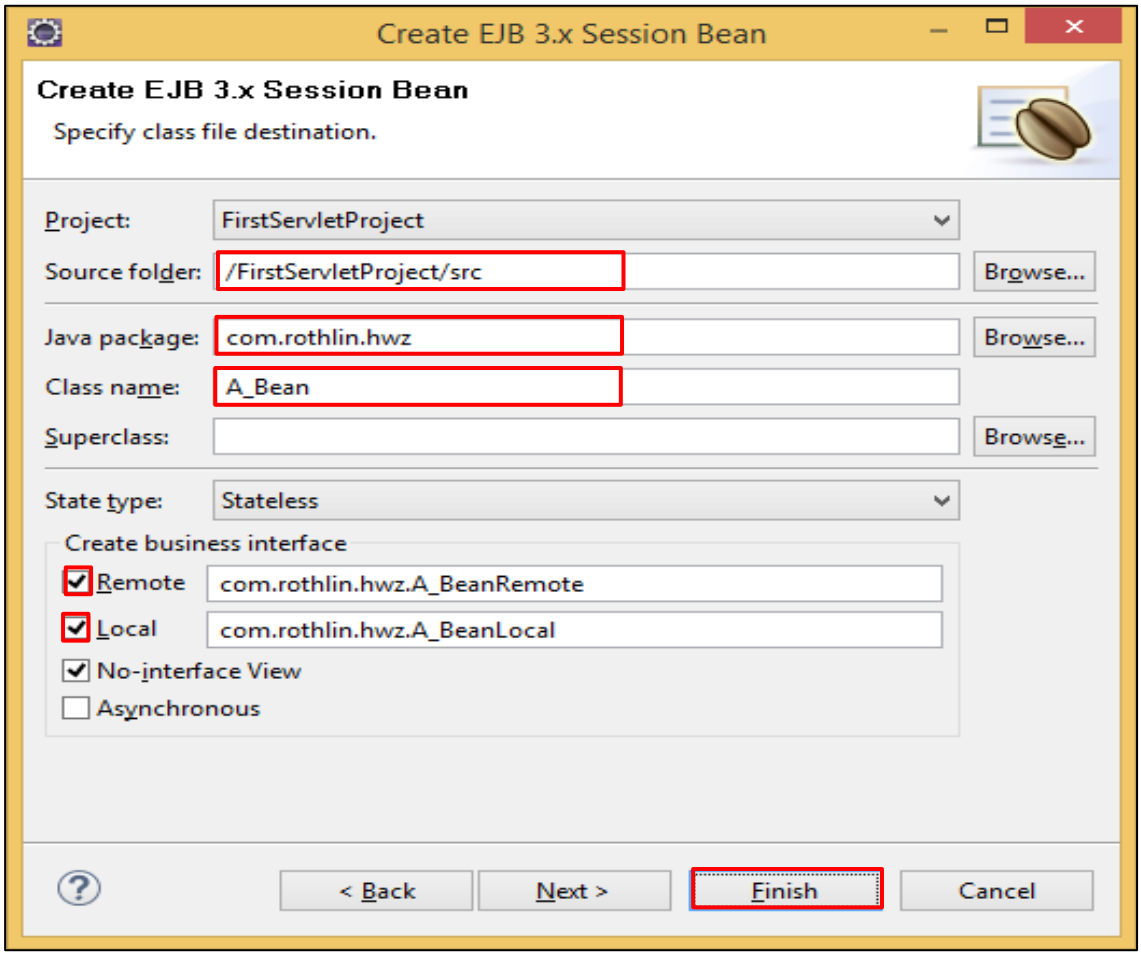
```
// PojoCallsRemoteEJBs belongs to 03_c_PojoCallsEJBsRemotely
// -------------------            -------------------------

package com.rothlin.hwz;

import javax.naming.InitialContext;

public class PojoCallsRemoteEJBs {

        private static final String jndiHost                        = "127.0.0.1";
        private static final String jndiPort                        = "3700";
        private static final String jndiName__EJB_A_StatelessRemote =
"java:global/03_a_EJBs_stateful_stateless/EJB_A_Stateless!com.rothlin.hwz.EJB_A_StatelessRemote";
        private static final String jndiName__EJB_B_StatefulRemote  =
"java:global/03_a_EJBs_stateful_stateless/EJB_B_Stateful!com.rothlin.hwz.EJB_B_StatefulRemote";

        private EJB_A_StatelessRemote ejbA = null;
        private EJB_B_StatefulRemote  ejbB = null;

        public PojoCallsRemoteEJBs() {
                System.out.println("PojoCallsBruchEJB:: ctr() is called!");
                System.setProperty("org.omg.CORBA.ORBInitialHost", jndiHost);
                        try {
                                InitialContext ctx = new InitialContext();
                                ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", jndiHost);
                                ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", jndiPort);
                                ejbA = (EJB_A_StatelessRemote) ctx.lookup(jndiName__EJB_A_StatelessRemote);
                                ejbB = (EJB_B_StatefulRemote)  ctx.lookup(jndiName__EJB_B_StatefulRemote);
                        }
                        catch (Exception ex) {
                                System.out.println("Could not get remote-object bean!" + ex);
                        }
        }

        public void runClient() {
                        System.out.println("PojoCallsRemoteEJBs has been started!");
                        System.out.println("ejbA.pingRemote(): " + ejbA.pingRemote());
                        System.out.println("ejbB.pingRemote(): " + ejbB.pingRemote());
        }

        public static void main(String[] args) {
                PojoCallsRemoteEJBs cli = new PojoCallsRemoteEJBs();
                try {
                        cli.runClient();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

Save and run it.



Create Bean to get called from Servlet / JSP in the "FirsServletProject"

```
File -> New -> Other -> EJB -> Session Bean (EJB 3.x)
```



Code for bean:

A_Bean.java

```
package com.rothlin.hwz;

import javax.ejb.LocalBean;
import javax.ejb.Stateless;


@Stateless
@LocalBean
public class A_Bean implements A_BeanRemote, A_BeanLocal {

        private int counter;

        public A_Bean() {
               System.out.println("Konstruktor von A_Bean aufgerufen!");
               counter = 10;
        }

        public String sayHello(String name) {
                counter++;
                return "HALLO    " + name + ": Der Counter ist: " + counter;
        }
}
```

A_BeanLocal.java

```
package com.rothlin.hwz;

import javax.ejb.Local;

@Local
public interface A_BeanLocal {
        public String sayHello(String name);
}
```

A_BeanRemote.java

```
package com.rothlin.hwz;

import javax.ejb.Remote;

@Remote
public interface A_BeanRemote {
        public String sayHello(String name);
}
```

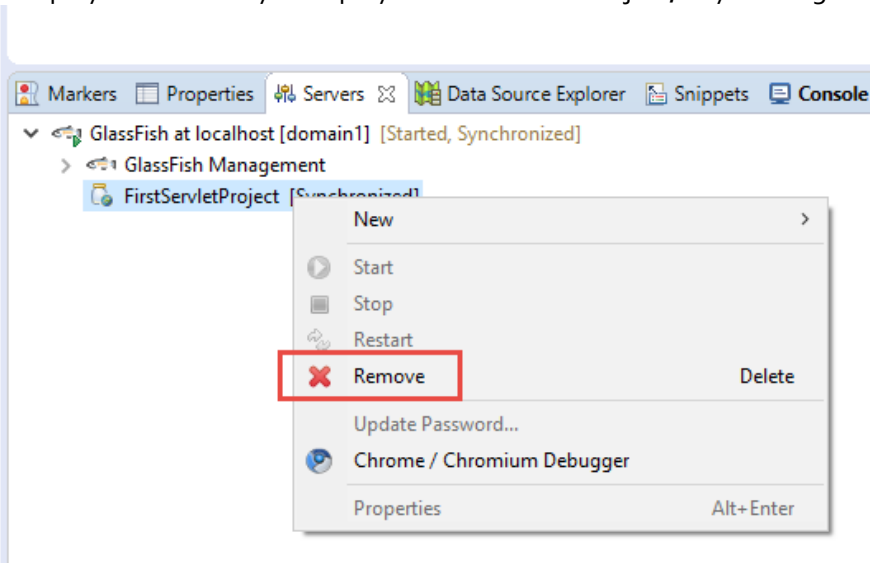
Call Bean-method from Servlet (same web-App):

Add the following code to **FirstServlet.java**:

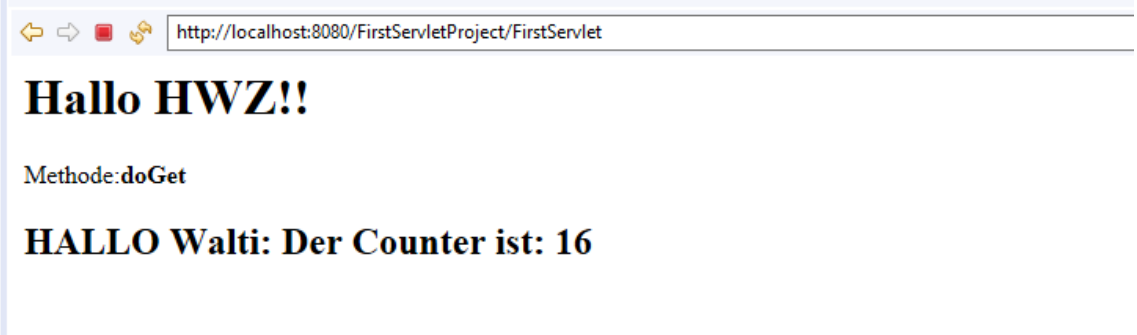
```
package com.rothlin.hwz;

import java.io.IOException;

import javax.ejb.EJB;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	@EJB
	A_Bean myBean;

	public FirstServlet() {
		super();
		System.out.println("FirstServlet ctr is called! Object created!!!");
	}

	public void init(ServletConfig servletConfig) throws ServletException {
		System.out.println("FirstServlet init() is called!");
	}

	public void destroy() {
		System.out.println("FirstServlet destroy() is called!");
	}

	private void serveRequest(HttpServletRequest request, HttpServletResponse
response, String methode)
		throws ServletException, IOException {
		response.setContentType("text/html");
		StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
		sb.append("<H1>Hallo HWZ!</H1> Methode:<B>" + methode + "</B><br/>\n");
		sb.append("<H2>" + myBean.sayHello("Walti")  + "</H2>\n");
		sb.append("</body>\n</html>\n");
		response.getWriter().print(sb);
	}

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		serveRequest(request, response, "doGet");
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		serveRequest(request, response, "doPost");
	}
}
```

Testing Servlet calling local bean

Remove deployment before you deploy the FirstServletProject, or you will get an internal server error.



After you deployed your project again, your servlet is running and if you test in a browser, you see



The counter starts with 11.

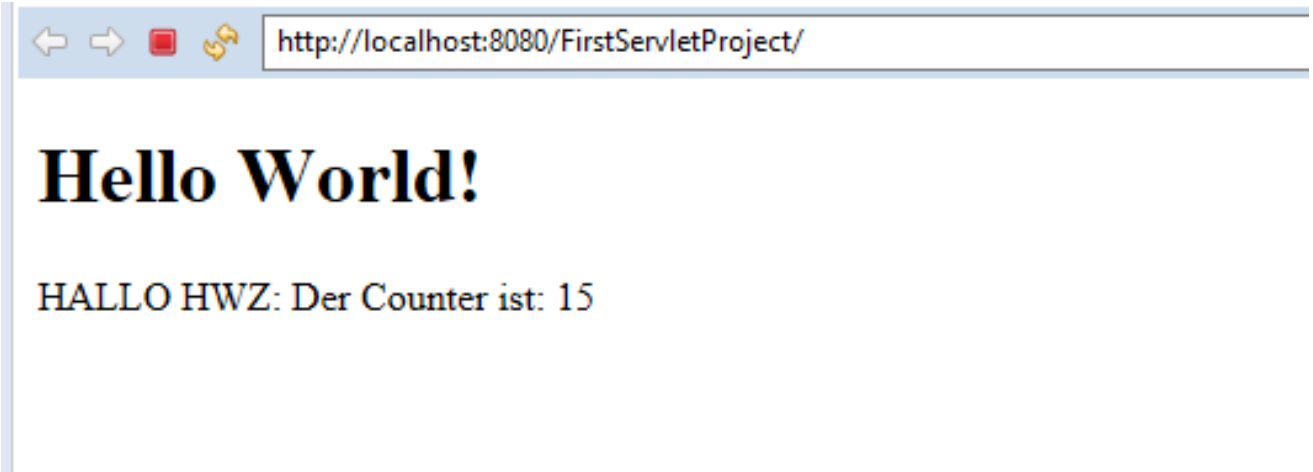
Call Bean-method from JSP (same web-App)

Add the following code to index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html4/loose.dtd">

<%@page          import ="com.rothlin.hwz.A_Bean" %>
<jsp:useBean     id     ="userBean" class="com.rothlin.hwz.A_Bean" scope="session"/>

<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>GlassFish JSP Page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <%= userBean.sayHello("HWZ") %><HR/>
  </body>
</html>
```

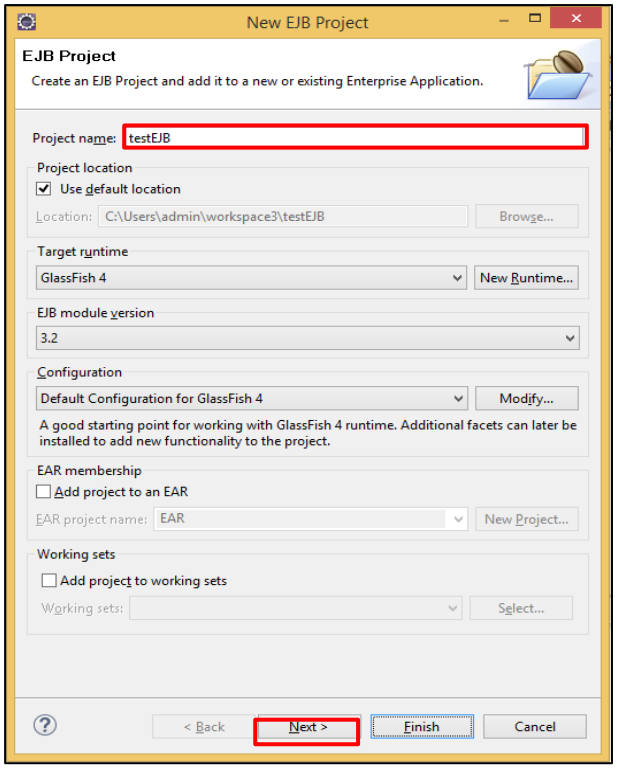
Testing JSP calling EJB local

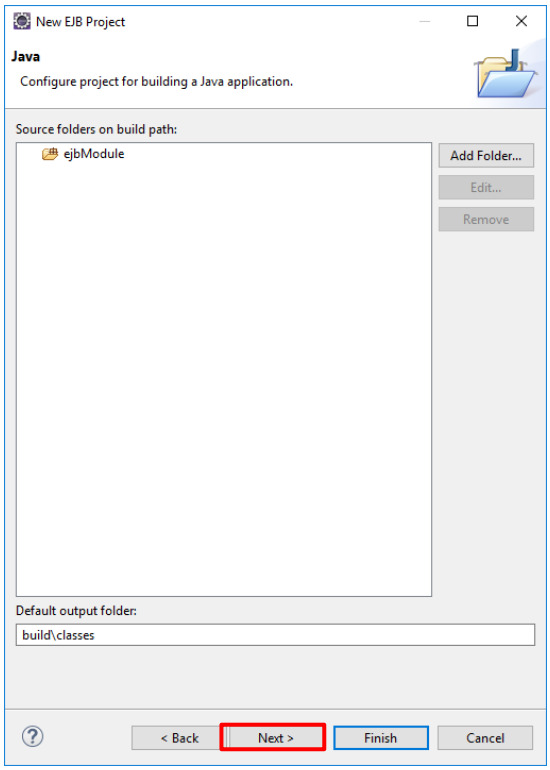
Example output after redeploy:

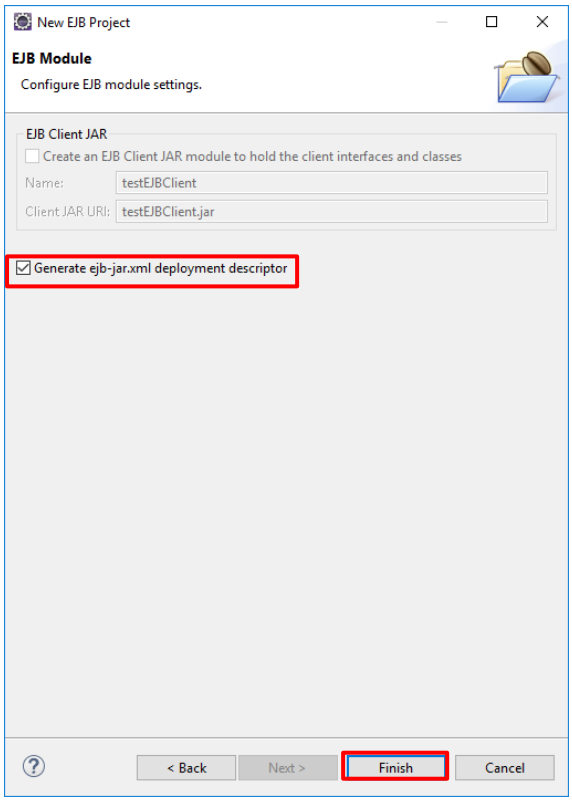


Create an EJB (Enterprise Java Bean) in another project (Remote)

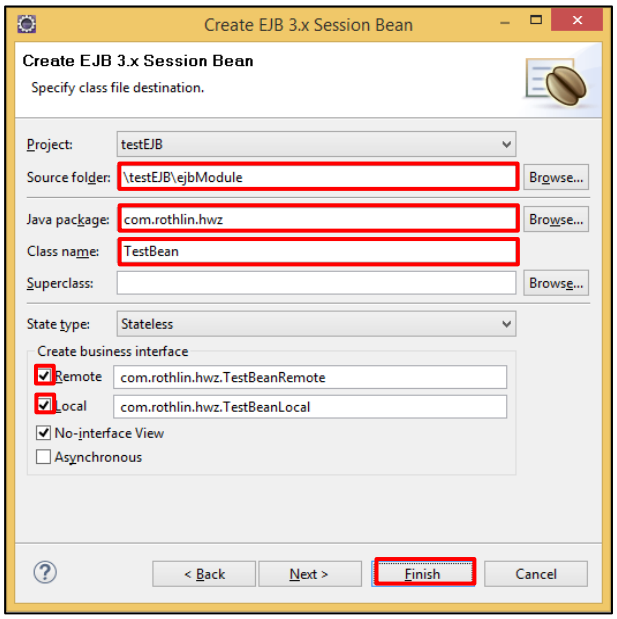
```
File -> New -> Other -> EJB Project
```







File -> New -> Other -> EJB -> Session Bean (EJB 3.x)



First EJB Code

Replace the existing code of the following classes with these:

TestBean.java:

```
package com.rothlin.hwz;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.LocalBean;
import javax.ejb.Stateless;


@Stateless
@LocalBean
public class TestBean implements TestBeanRemote, TestBeanLocal {

	@PostConstruct
	private void startup() {
	   System.out.println("TestBean: PostConstruct!!");
	}

	@PreDestroy
	public void cleanup() {
	   System.out.println("TestBean: cleanup!");
	}

	public String ping() {
	    return "TestBean V1.0";
	}

	public TestBean() {
	    System.out.println("TestBean: ctr called!");
	}
}
```

TestBeanLocal.java

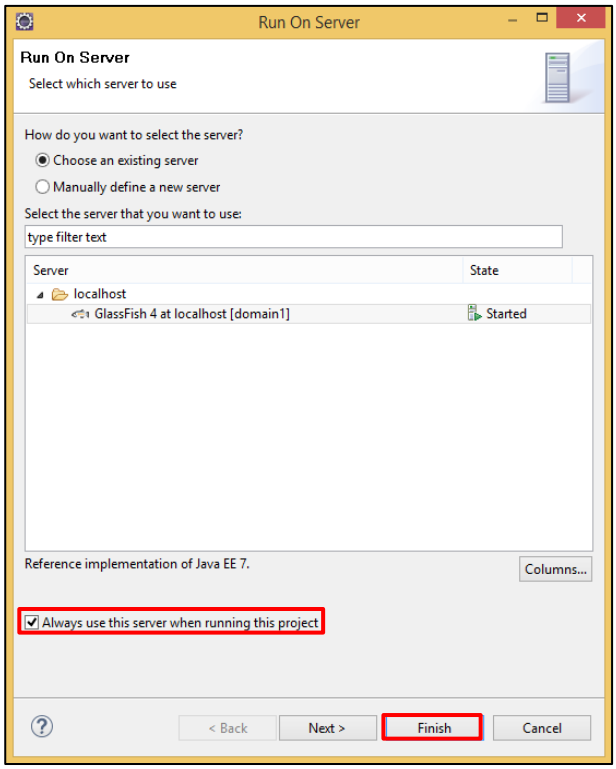
```
package com.rothlin.hwz;

import javax.ejb.Local;

@Local
public interface TestBeanLocal {
	public String ping();
}
```

TestBeanRemote.java

```
package com.rothlin.hwz;

import javax.ejb.Remote;

@Remote
public interface TestBeanRemote {
	public String ping();
}
```

Deploy EJB

Select the EJB Project and click on Run as -> Run on Server



A Portable JNDI name will be created (check serverlog):

```
2017-04-03T14:11:42.223+0200|Info: Portable JNDI names for EJB TestBean:
[java:global/testEJB/TestBean!com.rothlin.hwz.TestBeanLocal,
java:global/testEJB/TestBean!com.rothlin.hwz.TestBean,
java:global/testEJB/TestBean!com.rothlin.hwz.TestBeanRemote]
```

Call EJB Methode from "FirstServlet"

Modify the servlet-code as following:

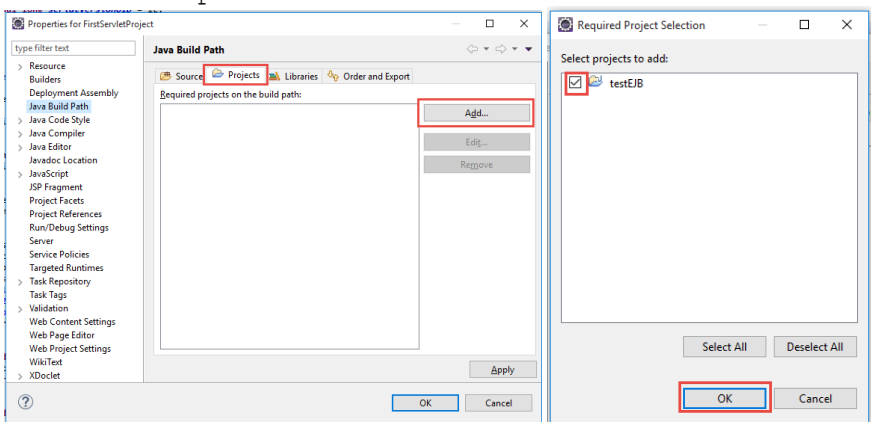
```
package com.rothlin.hwz;

import java.io.IOException;

import javax.ejb.EJB;
import javax.naming.InitialContext;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;

        @EJB
        A_Bean myBean;

        TestBeanRemote aTestBean = null;

        public FirstServlet() {
                super();
                System.out.println("FirstServlet ctr is called! Object created!!!");
        }

        public void init(ServletConfig servletConfig) throws ServletException {
                System.out.println("FirstServlet init() is called!");
                System.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
                try {
                        InitialContext ctx = new InitialContext();
                        ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
                        ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", "3700");
                        aTestBean = (TestBeanRemote) ctx.lookup
                             ("java:global/testEJB/TestBean!com.rothlin.hwz.TestBeanRemote");
                }
                catch (Exception ex) {
                    System.out.println("Couldn't create converter bean!" + ex);
                }
        }

        public void destroy() {
                System.out.println("FirstServlet destroy() is called!");
        }

        private void serveRequest(HttpServletRequest  request,
                                  HttpServletResponse response, String methode)
                throws ServletException, IOException {
                response.setContentType("text/html");
                StringBuffer sb = new StringBuffer("<html>\n<head>\n</head>\n<body>\n");
                sb.append("<H1>Hallo HWZ!</H1> Methode:<B>" + methode + "</B><br/>\n");
                sb.append("<H2>" + myBean.sayHello("Walti")  + "</H2>\n");
                sb.append("<H2>" + aTestBean.ping() + "</H2>\n");
                sb.append("</body>\n</html>\n");
                response.getWriter().print(sb);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
                serveRequest(request, response, "doGet");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
                serveRequest(request, response, "doPost");
        }
}
```

Fixen der Projektübergreifenden Interfaces

Um den Fehler des vermissten **TestBeanRemote** zu fixen, kopieren sie das gleichnamige File vom **testEJB**-Projekt in das **FirstServletProject**

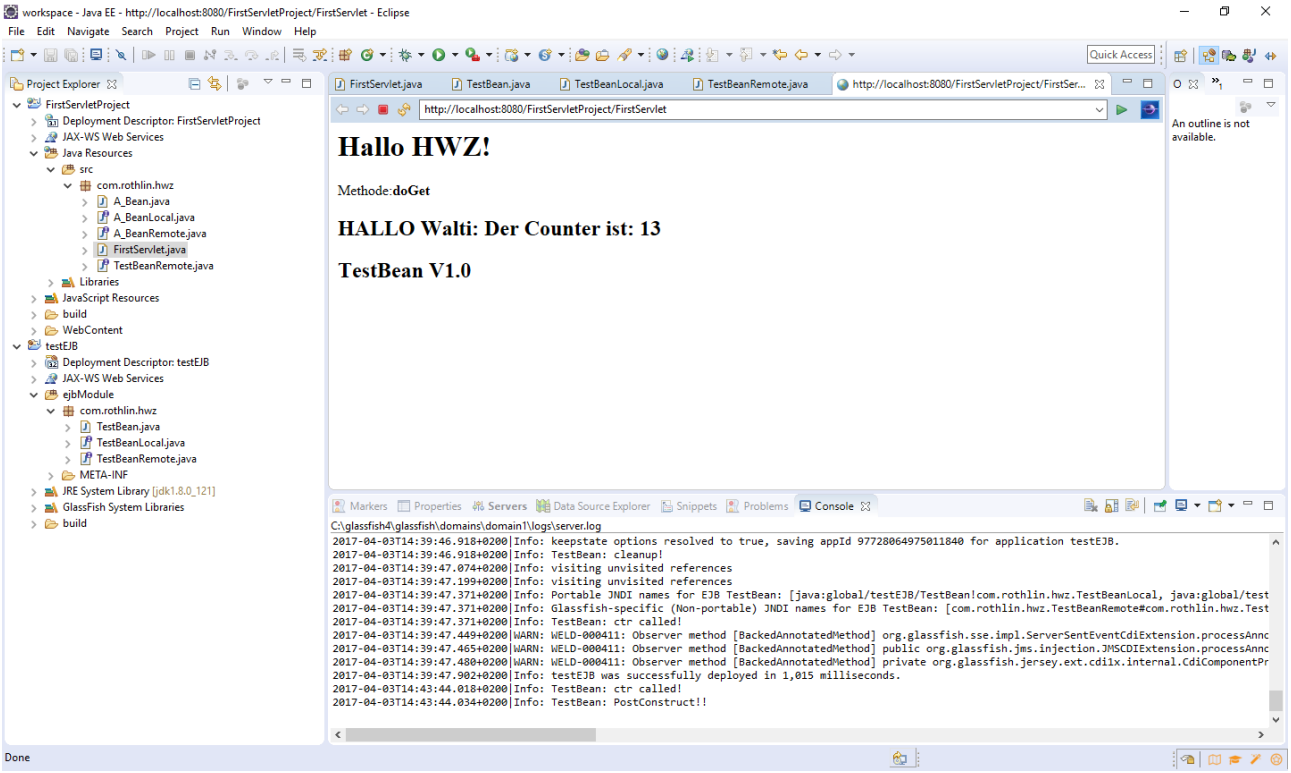
Es gibt auch die Möglichkeit die Resourcen im **testEJB** im **FirstServletProject** bekannt zu machen:
Add testEJB Project to FirstServletProject

```
File -> Properties -> Java Build Path
```



If the project testEJB is visible as build path, close the window.

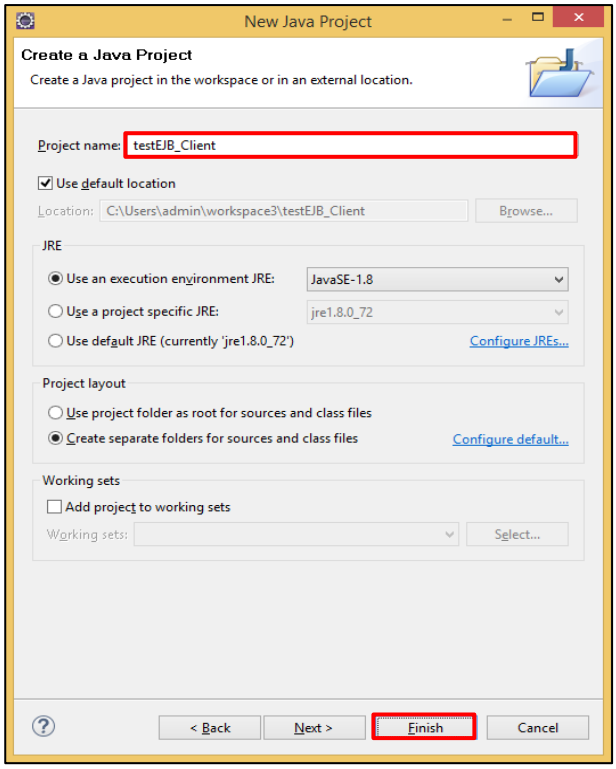
Testen des Servlets mit EJB local and remote call



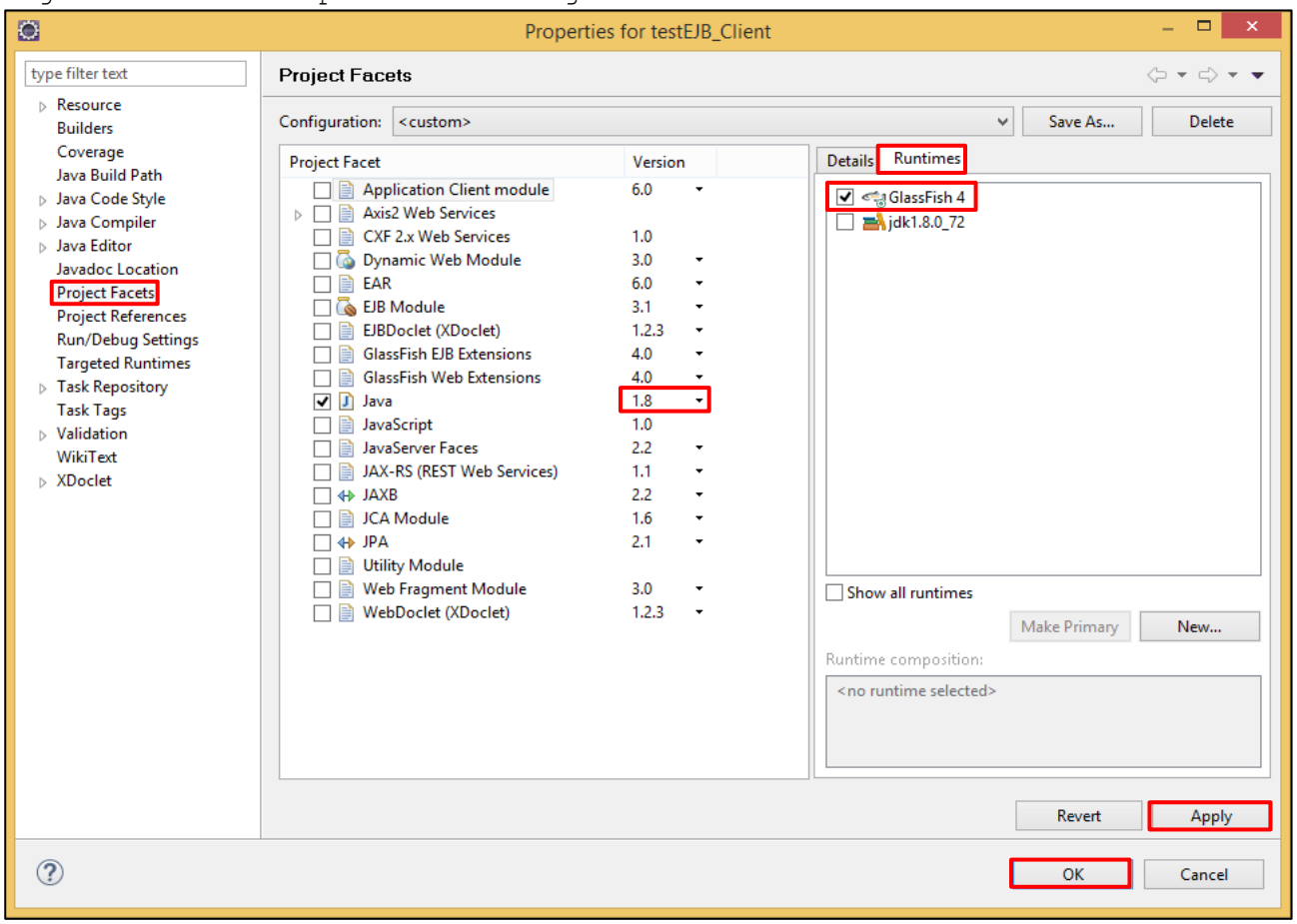
Wichtig: wird nun der Code des EJB geändert und deployed, muss auch das Servlet neu geladen werden (die init() function muss aufgerufen werden und eine neue Remote-Referenz via JNDI geholt werden). Am einfachsten macht man das, indem der Servlet Code geändert wird und wieder abgespeichert, so wird das Servlet ebenfalls neu deployed und init() aufgerufen.

Java-client calls EJB-Methode

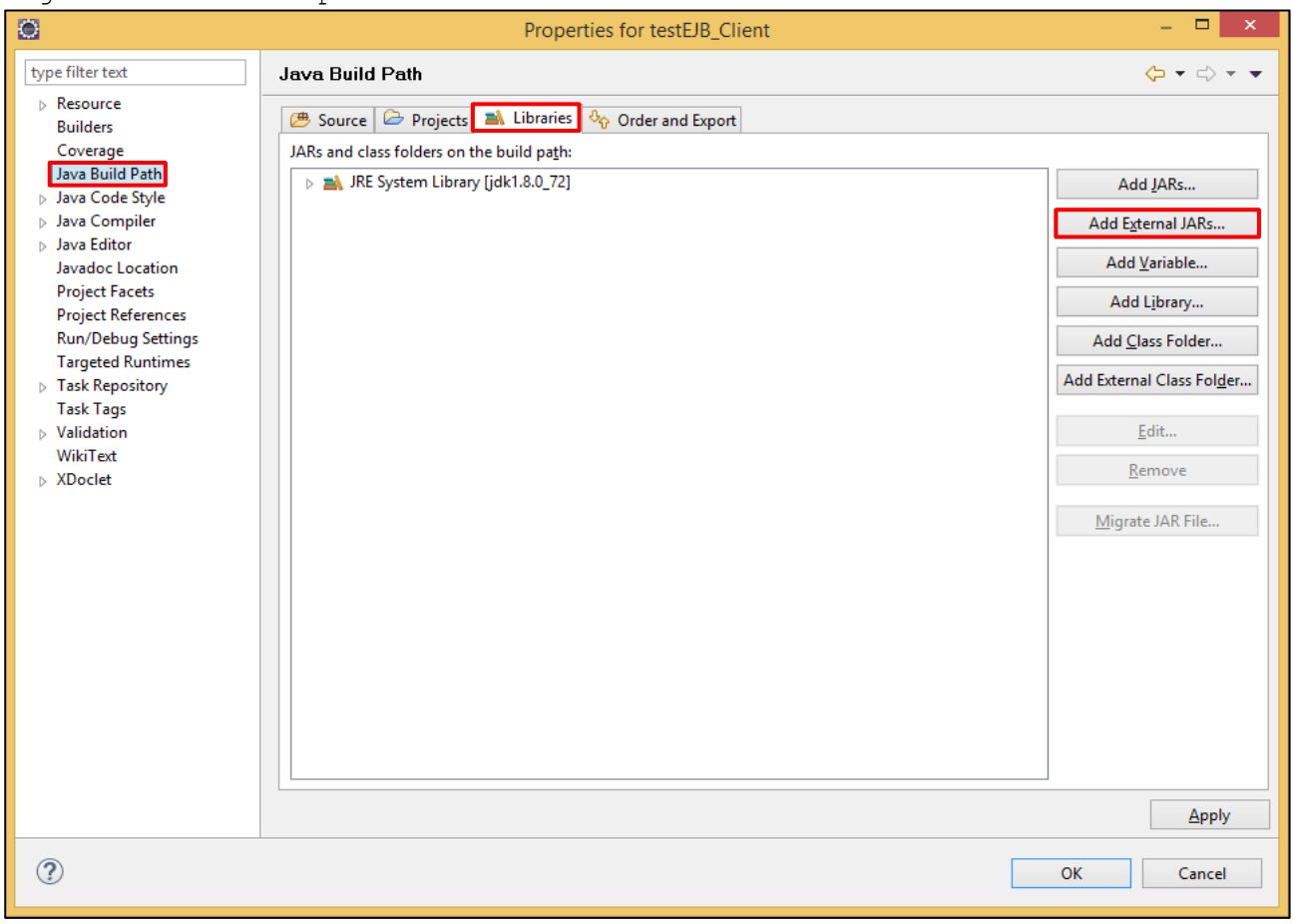
File -> New -> Other -> Java -> Java Project

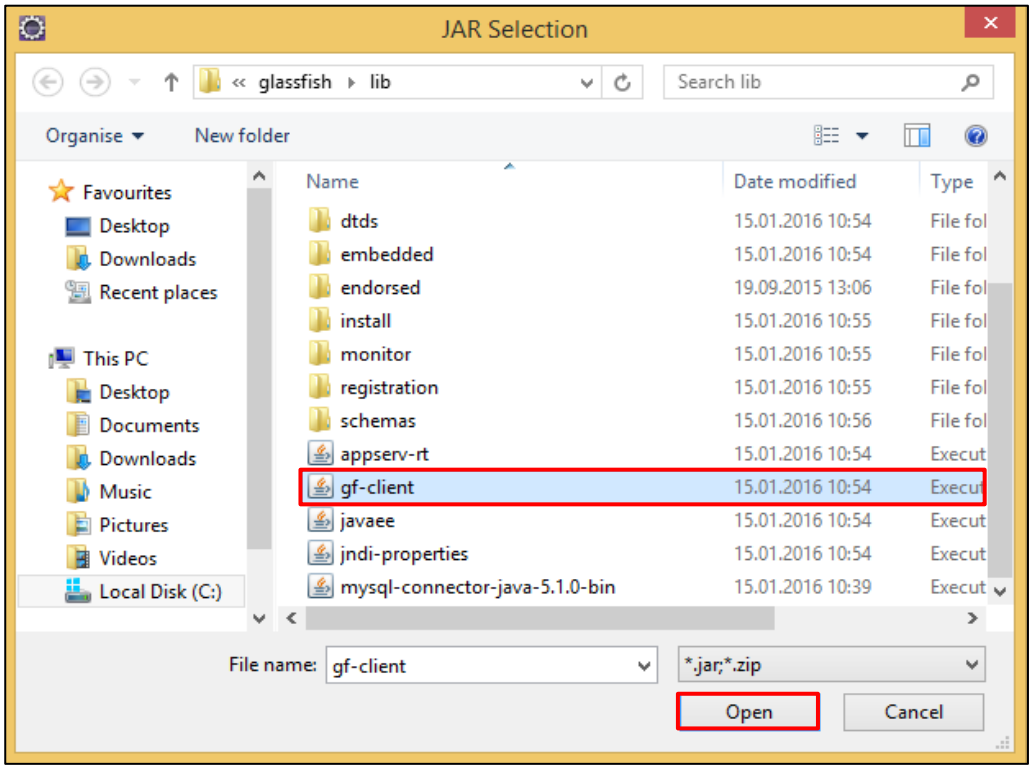


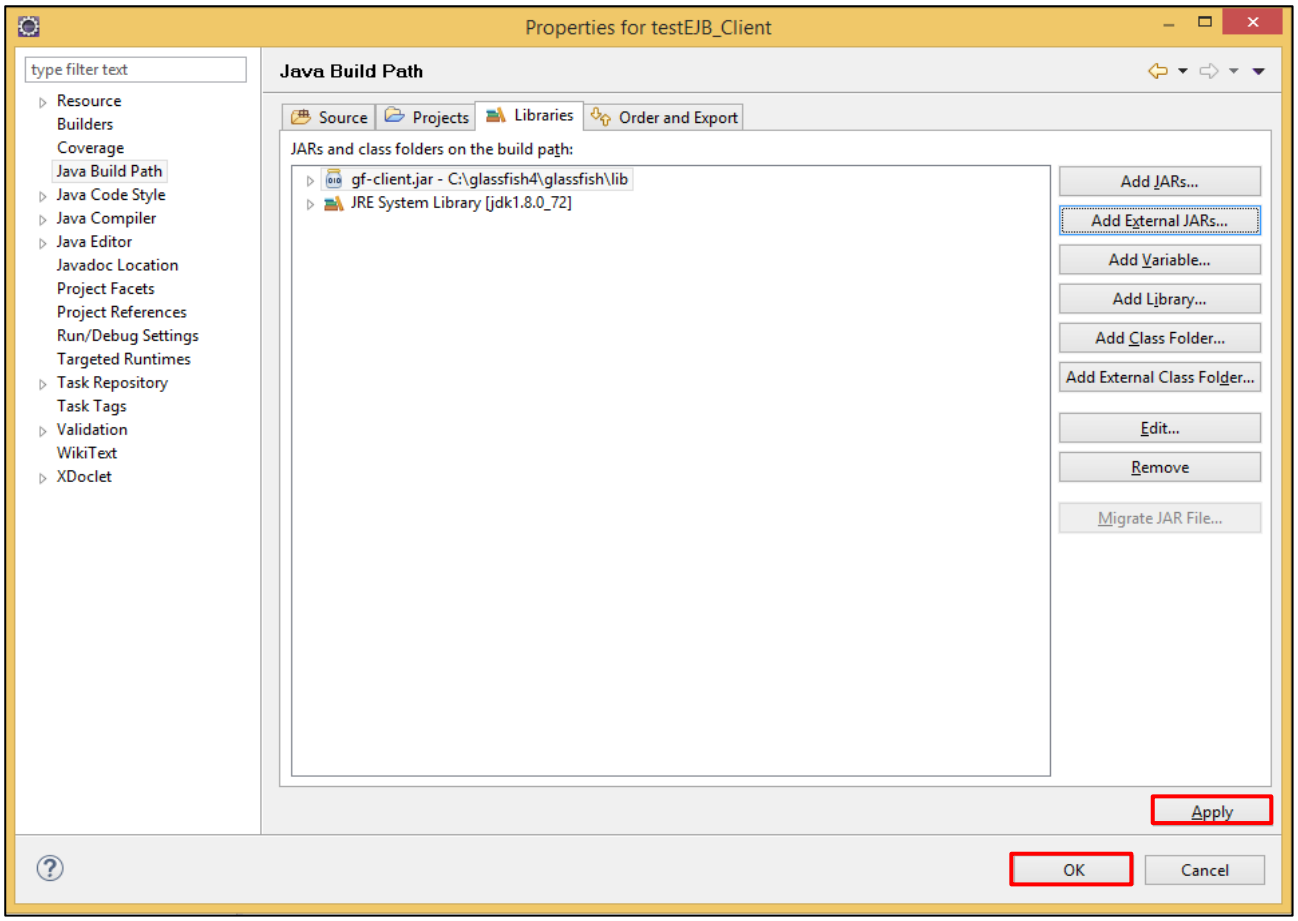
Right click -> Properties -> Project Facets -> Runtimes -> Glassfish 4



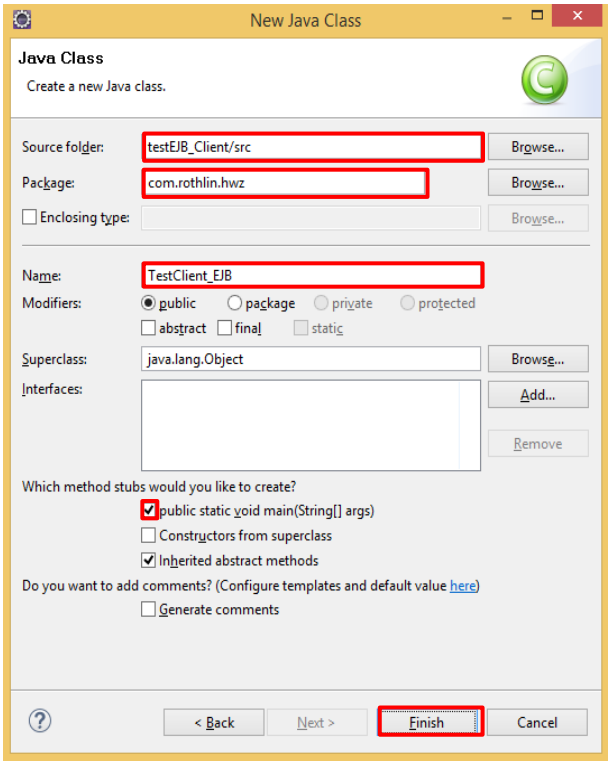
Right click -> Properties -> Java Build Path -> Libraries







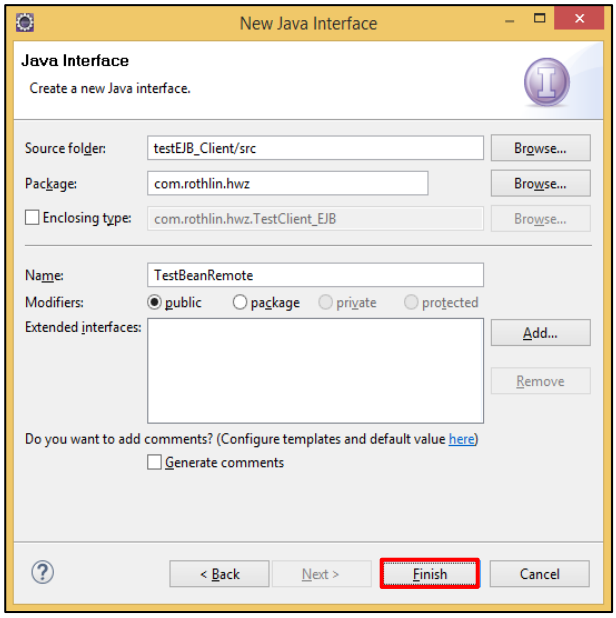
File -> New -> Other -> Java -> Class



TestClient_EJB.java

```
package com.rothlin.hwz;

import javax.naming.InitialContext;

public class TestClient_EJB {
	public void runTest() throws Exception {
		System.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
		InitialContext ctx = new InitialContext();

		/* calling stateless session bean */
		System.out.println("Calling TestBean (Session stateless)");
		ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
		ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", "3700");


		TestBeanRemote bean = (TestBeanRemote) ctx.lookup
				("java:global/testEJB/TestBean!com.rothlin.hwz.TestBeanRemote");

		System.out.println(bean.ping());

	}

	public static void main(String[] args) {
		TestClient_EJB cli = new TestClient_EJB();
		try {
			cli.runTest();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

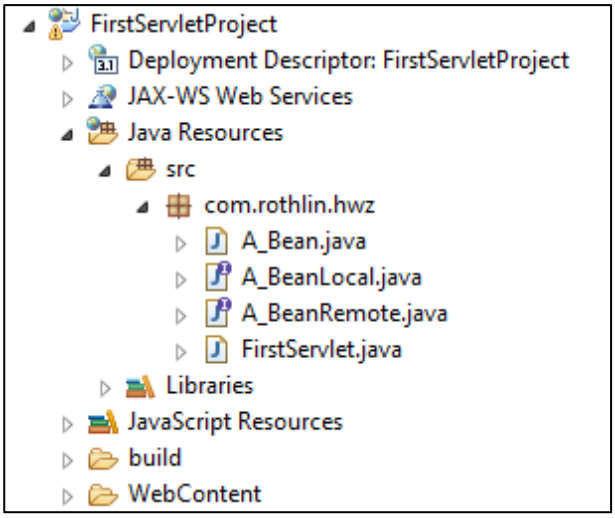
File -> New -> Other -> Java -> Java Interface

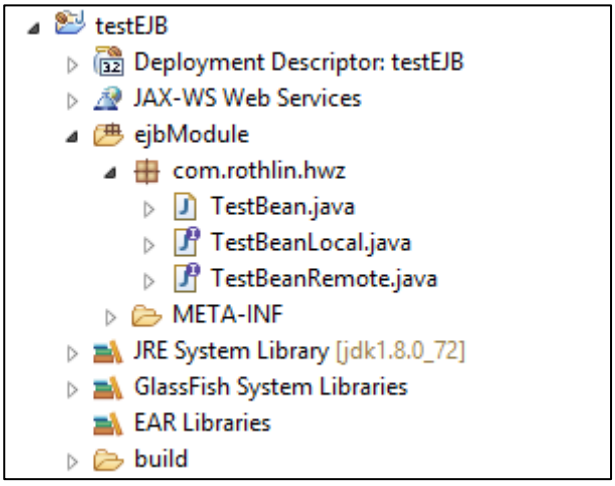


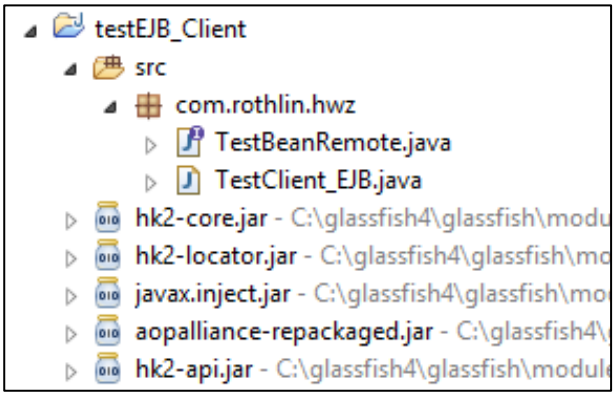
TestBeanRemote.java

```
package com.rothlin.hwz;

public interface TestBeanRemote {
	public String ping();
}
```

This is how the Package Explorer should look like:

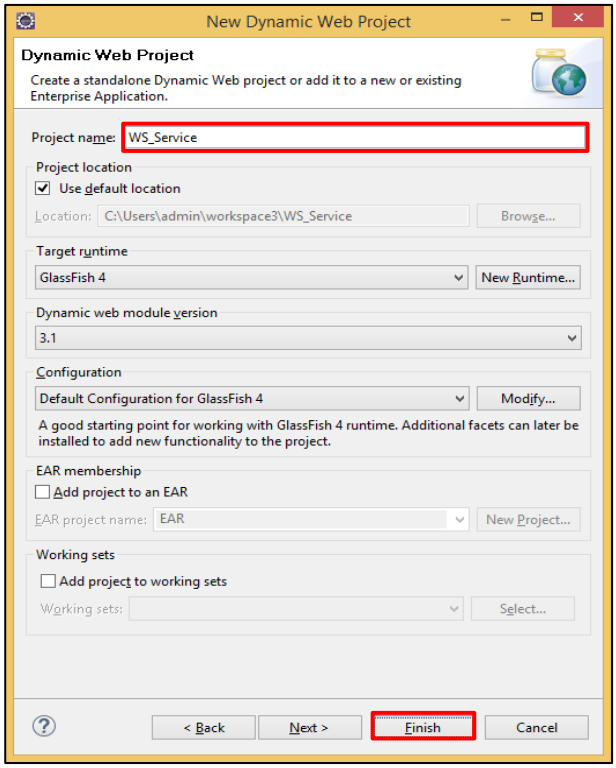




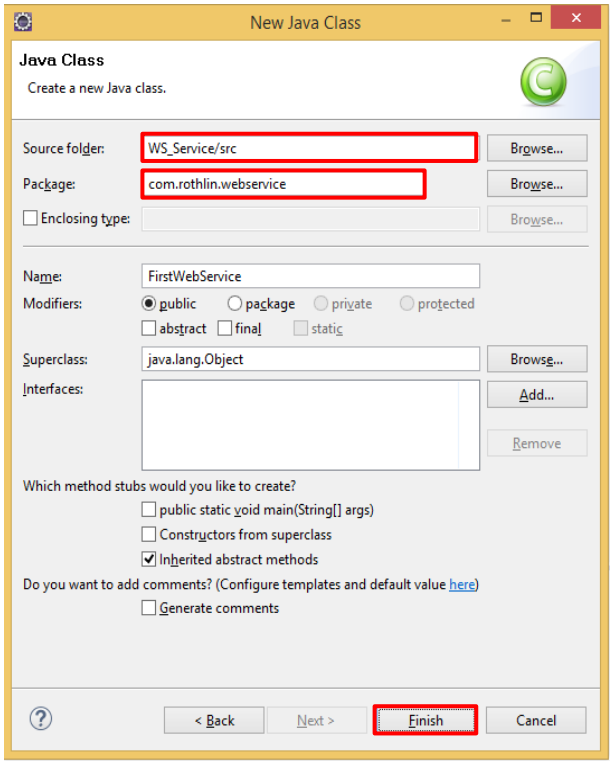


Create own WEB-Service

File -> New -> Other -> Web -> Dynamic Web Project



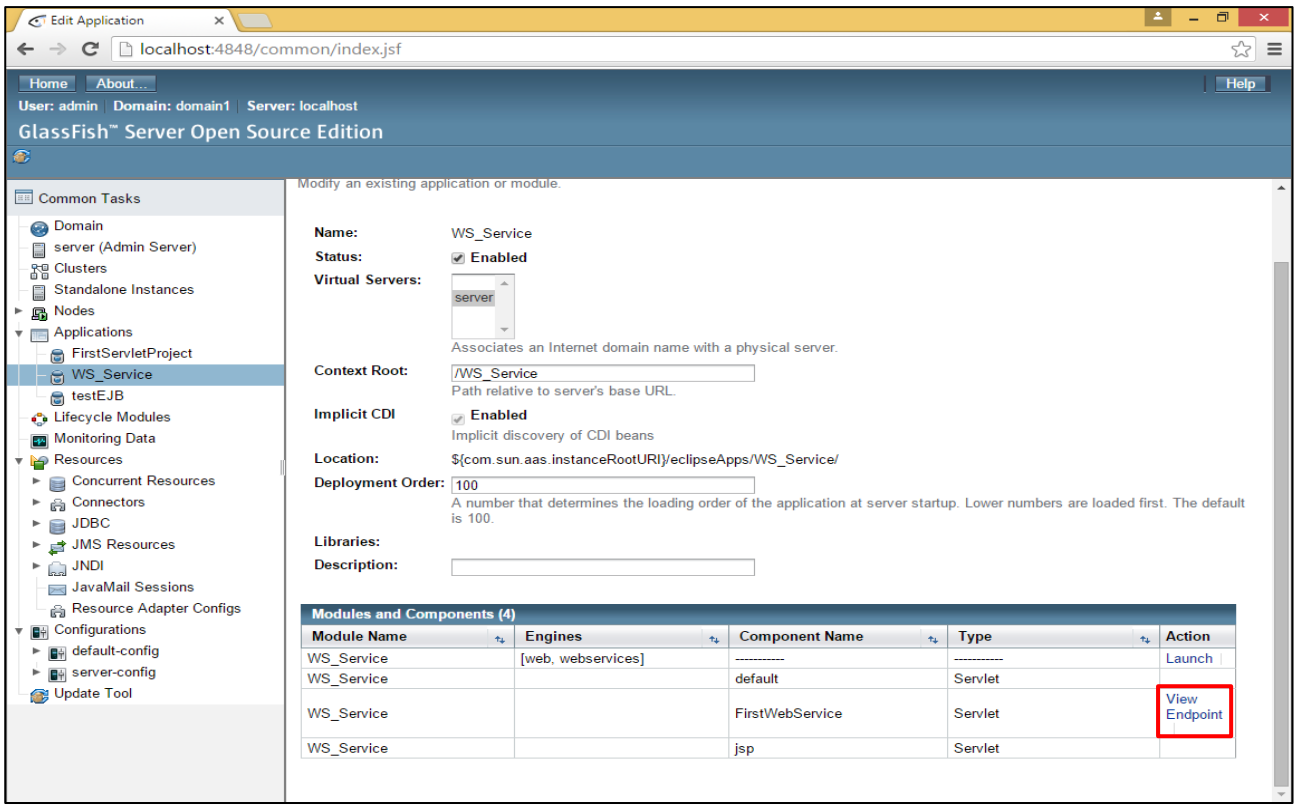
File -> New -> Other -> Java -> Class

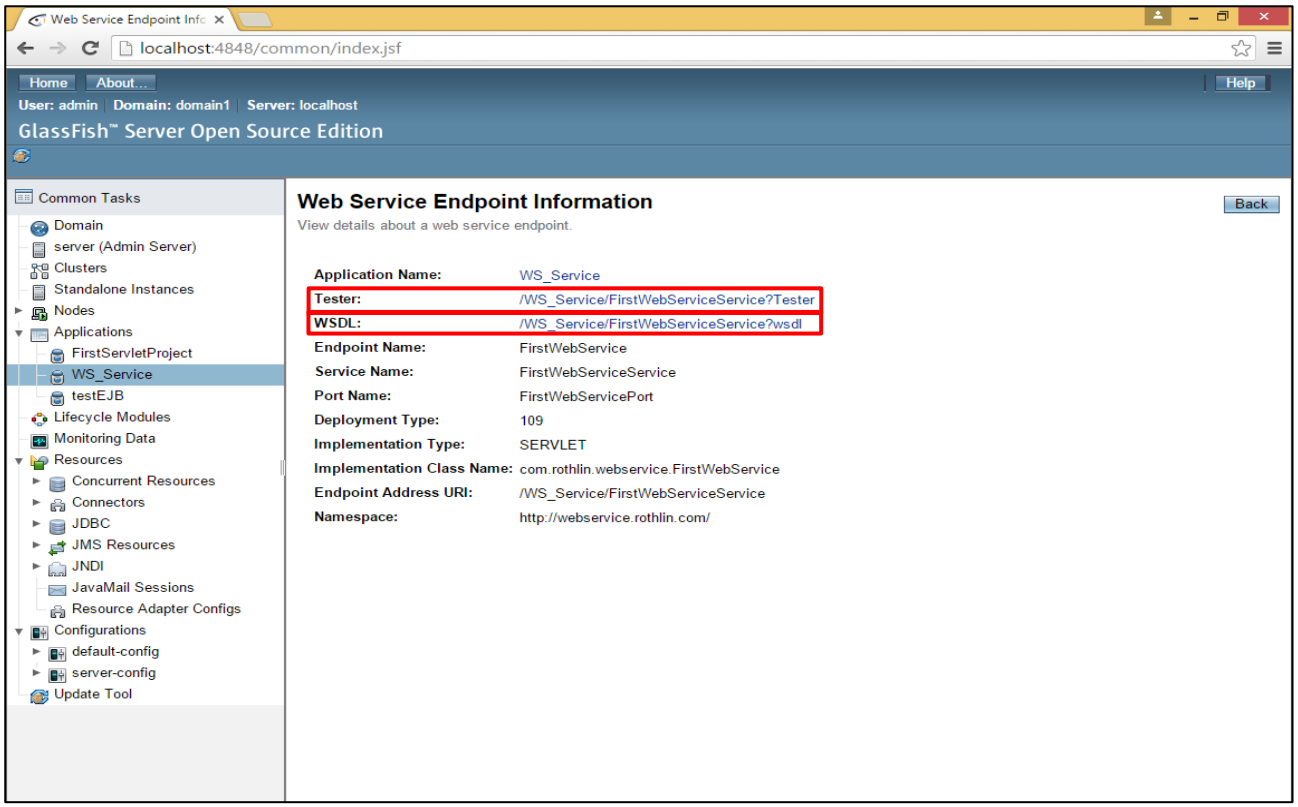


Add the following code:

```
package com.rothlin.webservice;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class FirstWebService {

        // Methodes in a web-Service are called Operations
        // Operation names can not be overloaded (
        @WebMethod
        public String sayHello() {
                return "Hello HWZ";
        }

        @WebMethod() // ctrl-space shows you the options
        public String sayHelloName(String name) {
                return "Hello " + name;
        }

        @WebMethod(exclude=false)
        public String sayAnredeName(String anrede, String name) {
                return anrede + " " + name;
        }
}
```

Run java class on Server and check Admin Console



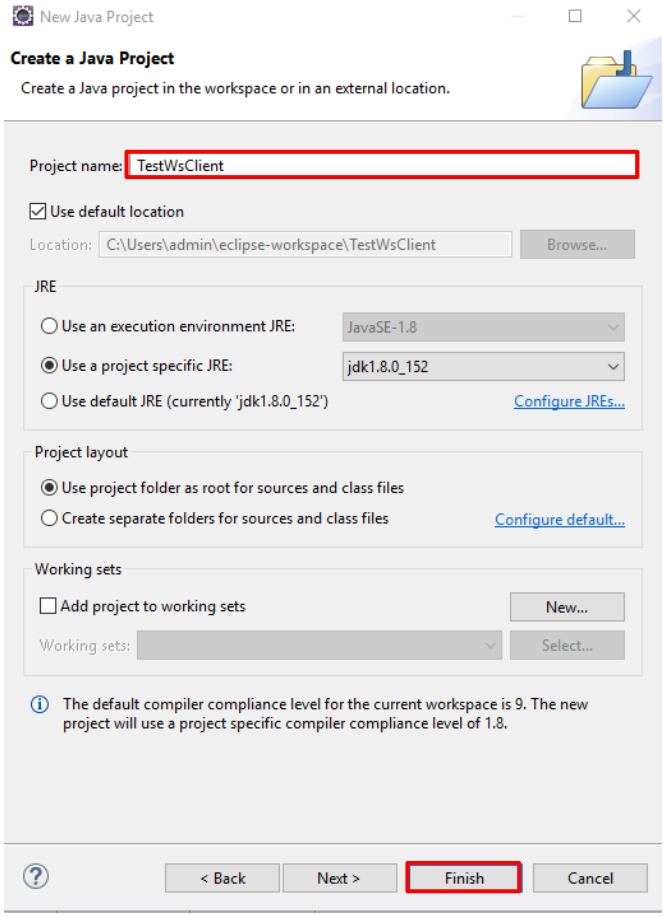


Tester — To test the operation. When you want to test the Web-Service without writing a client, click on the Tester-Link which is given in the Web Service Endpoint Information.

WSDL — To get the generated WSDL.

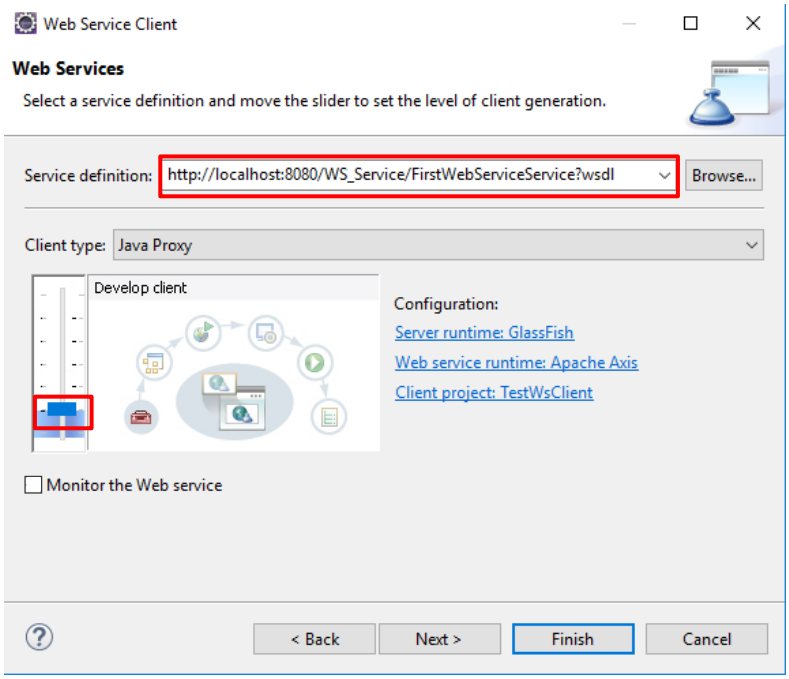
Create Java-Client calling WEB-Service

File -> New -> Java Project



Create the stub-classes from wsdl-file:

New -> Other… -> Web Service -> Web Service Client



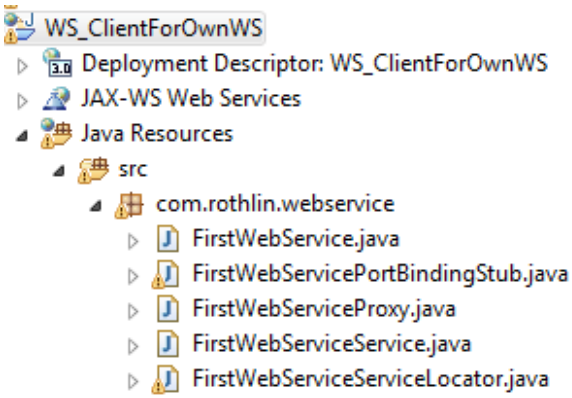
http://localhost:8080/WS_Service/FirstWebServiceService?wsdl

Other way to create stub-classes:

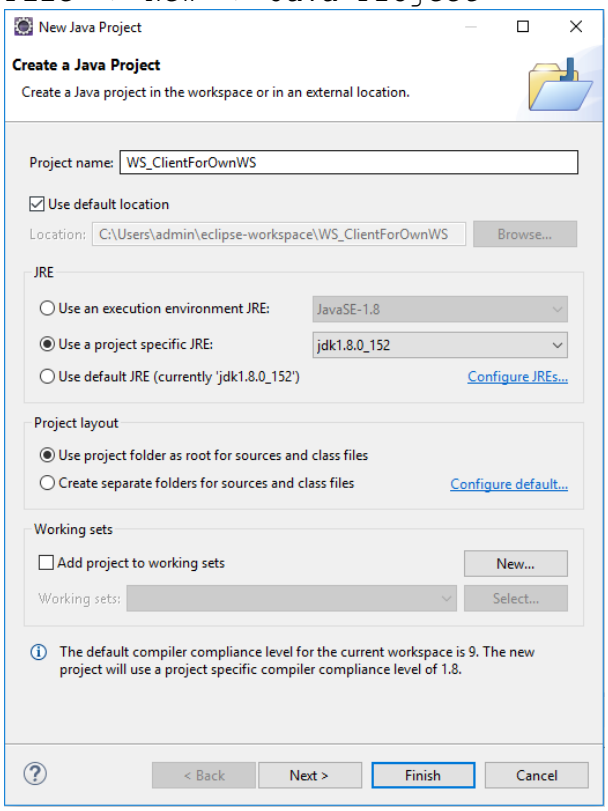
Generate Client Stub from WSDL in dos-prompt:

```
mkdir sei
cd sei
mkdir src
wsimport -keep -s src http://DESKTOP-0ORFPM9:8080/04_a_Webservice_Simple/FirstWebServiceService?wsdl
```

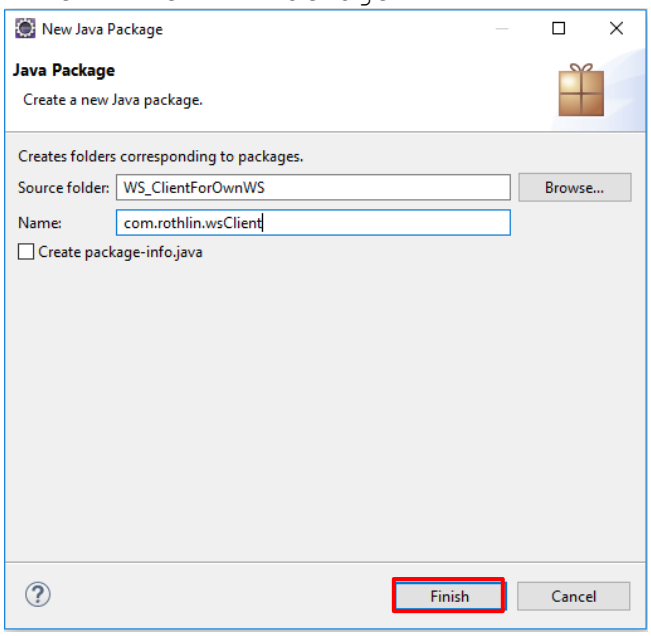
Created stub-classes:



File -> New -> Java Project

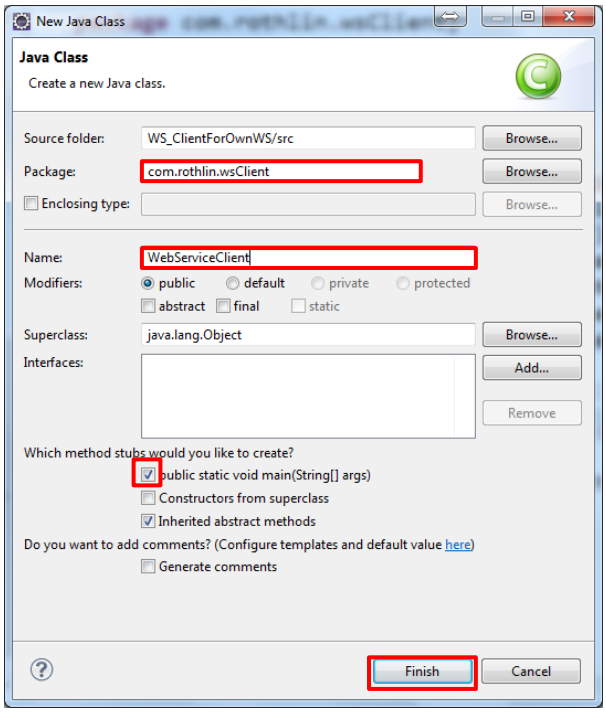


File -> New -> Package



Create client:

File -> New -> Class




```
package com.rothlin.wsClient;

import java.rmi.RemoteException;
import com.rothlin.webservice.FirstWebServiceProxy;

public class WebServiceClient {

	public static void main(String[] args) throws RemoteException {
		FirstWebServiceProxy ws = new FirstWebServiceProxy();
		System.out.println(ws.sayHello());
		System.out.println(ws.sayHelloName("BWI-14"));
		System.out.println(ws.sayAnredeName("Herr","Peter"));
		System.out.println(ws.calcCircleArea(5.0));
		System.out.println(ws.calcCircleUmfang(5.0));
	}
}
```

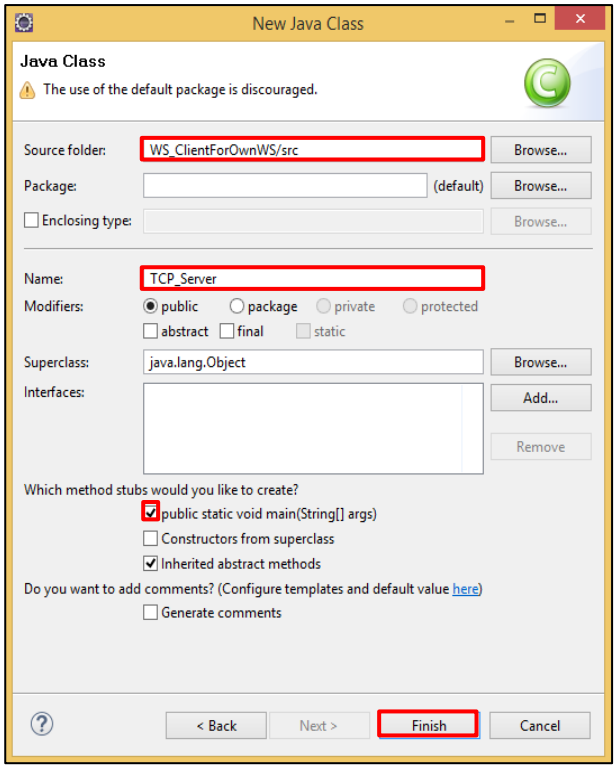
Create a Java Client which calls a public Web-Service

Used Web-Service (Converts an IP-address to a country name):
http://www.webservicex.net/geoipservice.asmx?WSDL

```
package com.rothlin.wsClient;

import net.webservicex.GeoIP;
import net.webservicex.GeoIPService;
import net.webservicex.GeoIPServiceSoap;


public class WebServiceClient {

        static public String getCountryNameForIP(String ipAddress) {
                // from WSDL: <wsdl:service name="GeoIPService">
                GeoIPService ipService = new GeoIPService();

                // from WSDL: <wsdl:port name="GeoIPServiceSoap">
                GeoIPServiceSoap geoIPServiceSoap = ipService.getGeoIPServiceSoap();

                GeoIP geoIp = geoIPServiceSoap.getGeoIP(ipAddress);
                return geoIp.getCountryName();
        }

        public static void main(String[] args) {
                if (args.length != 1) {
                        System.out.println("You need to pass one IP address!");
                        System.out.println("\nSome test cases:");
                        String ipAddress = "193.134.255.237";
                        System.out.println(ipAddress + " --> " + getCountryNameForIP(ipAddress));
                        ipAddress = "212.58.246.103";
                        System.out.println(ipAddress + " --> " + getCountryNameForIP(ipAddress));

                } else {
                        String ipAddress = args[0];
                        System.out.println(ipAddress + " --> " + getCountryNameForIP(ipAddress));

                }

        }

}
```

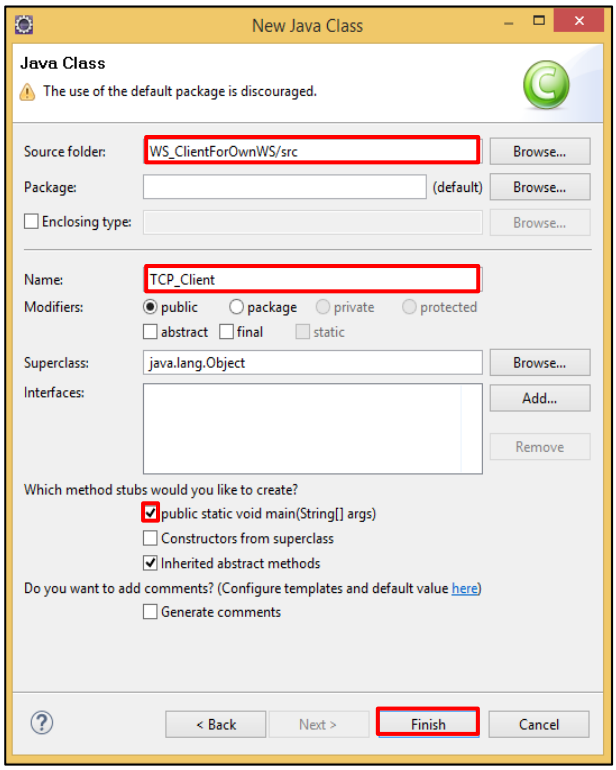
Create TCP/IP Server-Client

File -> New -> Other -> Java -> Class




```
import java.io.*;
import java.net.*;

class TCP_Server {
   public static void main(String argv[]) throws Exception {
         String clientSentence;
         String upperSentence ;
         ServerSocket welcomeSocket = new ServerSocket(6789);

         System.out.println("TCP/IP Server started!");
         while(true) {
            System.out.println("TCP/IP Server waiting to be called....");
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient =
               new BufferedReader(
                 new InputStreamReader(connectionSocket.getInputStream()));

            DataOutputStream outToClient =
               new DataOutputStream(connectionSocket.getOutputStream());

            clientSentence = inFromClient.readLine();
            System.out.println("Received from Client: " + clientSentence);
            upperSentence  = clientSentence.toUpperCase() + '\n';
            System.out.println("Sent back to Client : " + upperSentence );
            outToClient.writeBytes(upperSentence);
         }
      }
}
```

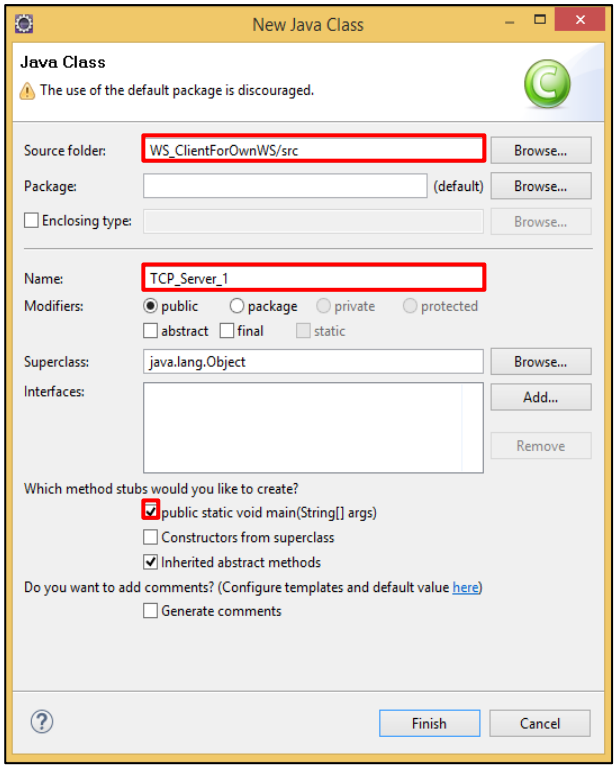
File -> New -> Other -> Java -> Class



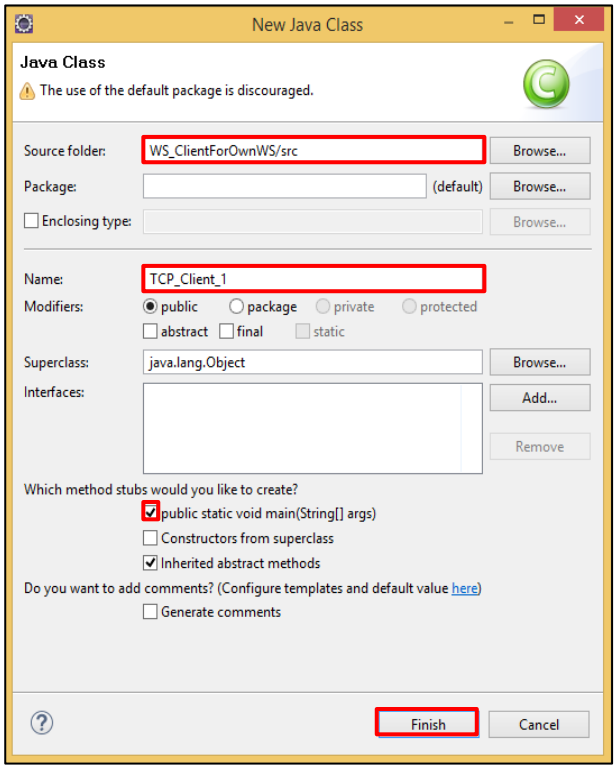

```java
import java.io.*;
import java.net.*;

class TCP_Client {
        public static void main(String argv[]) throws Exception {
                String sentence;
                String modifiedSentence;

                BufferedReader inFromUser =
                      new BufferedReader(
                        new InputStreamReader(System.in));
                Socket clientSocket = new Socket("localhost", 6789);

                DataOutputStream outToServer =
                      new DataOutputStream(clientSocket.getOutputStream());

                BufferedReader inFromServer =
                      new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));

                System.out.print("Enter a String      : ");
                sentence = inFromUser.readLine();

                System.out.println("Sent to SERVER      : " + sentence);

                outToServer.writeBytes(sentence + '\n');
                modifiedSentence = inFromServer.readLine();

                System.out.println("Received from SERVER: " + modifiedSentence);
                clientSocket.close();
                System.out.println("Client terminated!!!!!");
        }
}
```

Create TCP/IP Server-Client advanced

File -> New -> Other -> Java -> Class



File -> New -> Other -> Java -> Class



Add the following code to TCP_Server_1:

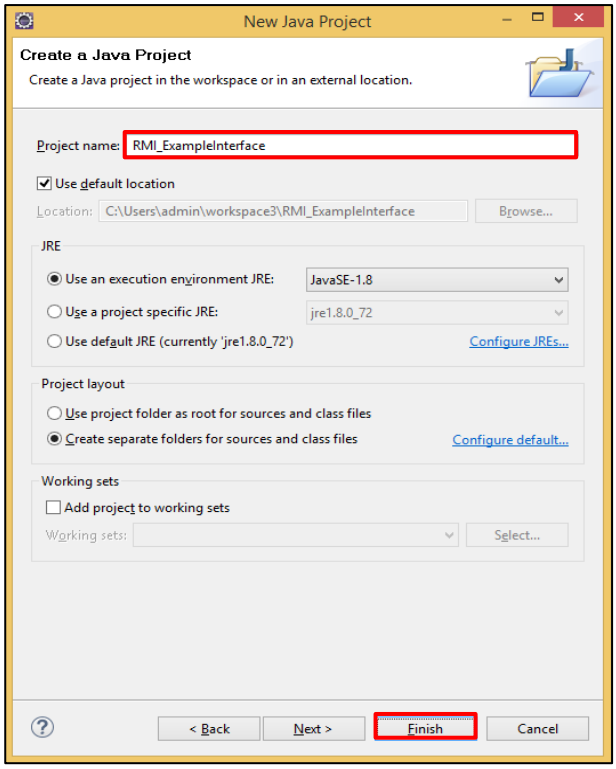
```
import java.io.*;
import java.net.*;


public class TCP_Server_1 {
   private static String businessFct_A(String msg) {
     return msg.toUpperCase() + '\n';
   }

   private static String businessFct_B(String msg) {
     return msg.toLowerCase() + '\n';
   }

   public static void main(String argv[]) throws Exception {
     String clientMsg;
     String replyMsg;
     ServerSocket welcomeSocket = new ServerSocket(6789);

     System.out.println("TCP/IP Server started!");
     while(true) {
        System.out.println("TCP/IP Server waiting to be called....");
        Socket connectionSocket = welcomeSocket.accept();
        BufferedReader inFromClient =
               new BufferedReader(
                  new InputStreamReader(connectionSocket.getInputStream()));
        DataOutputStream outToClient =
                  new DataOutputStream(connectionSocket.getOutputStream());

        clientMsg = inFromClient.readLine();
        System.out.println("Received from Client: " + clientMsg);
        String[] msgParts = clientMsg.split(":");
        String fctName   = msgParts[0];
        String stringMsg = msgParts[1];
        System.out.println("    Function : " + fctName);
        System.out.println("    Parameter: " + stringMsg);

        if (fctName.equals("toUpper")) {
                replyMsg  = businessFct_A(stringMsg);
        } else if (fctName.equals("toLower")) {
                replyMsg  = businessFct_B(stringMsg);
        } else {
                replyMsg  = "ERROR: Unknown Function " + fctName + '\n';
        }
        System.out.println("Sent back to Client : " + replyMsg );
        outToClient.writeBytes(replyMsg);
     }
  }
}
```

Add the following code to TCP_Client_1:

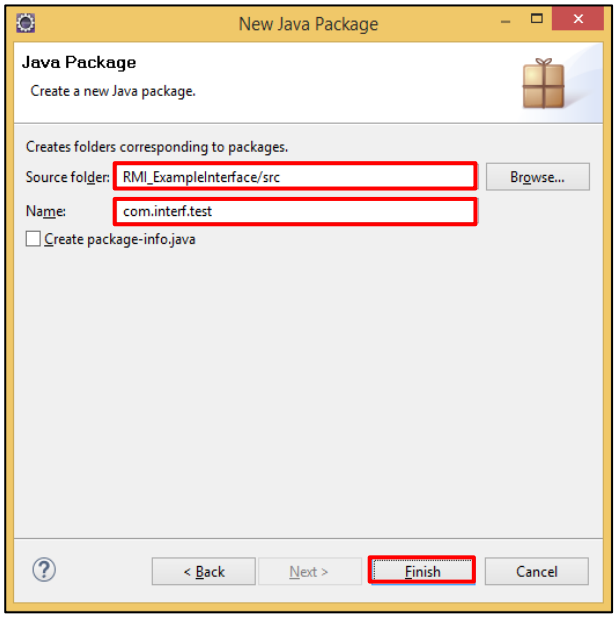
```
import java.io.*;
import java.net.*;

public class TCP_Client_1 {
  public static void main(String argv[]) throws Exception {
        String sentence;
        String fctName;
        String modifiedSentence;

        BufferedReader inFromUser = new BufferedReader(
                new InputStreamReader(System.in));

        Socket clientSocket = new Socket("localhost", 6789);

        DataOutputStream outToServer =
                 new DataOutputStream(clientSocket.getOutputStream());

        BufferedReader inFromServer =
                new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));

        System.out.print("Enter function name : ");
        fctName = inFromUser.readLine();
        System.out.print("Enter a String      : ");
        sentence = inFromUser.readLine();

        System.out.println("Sent to SERVER      : " + fctName + ":" + sentence);
        outToServer.writeBytes(fctName + ":" + sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("Received from SERVER: " + modifiedSentence);
      clientSocket.close();
        System.out.println("Client terminated!!!!!");
   }
}
```

Create RMI Client-Server

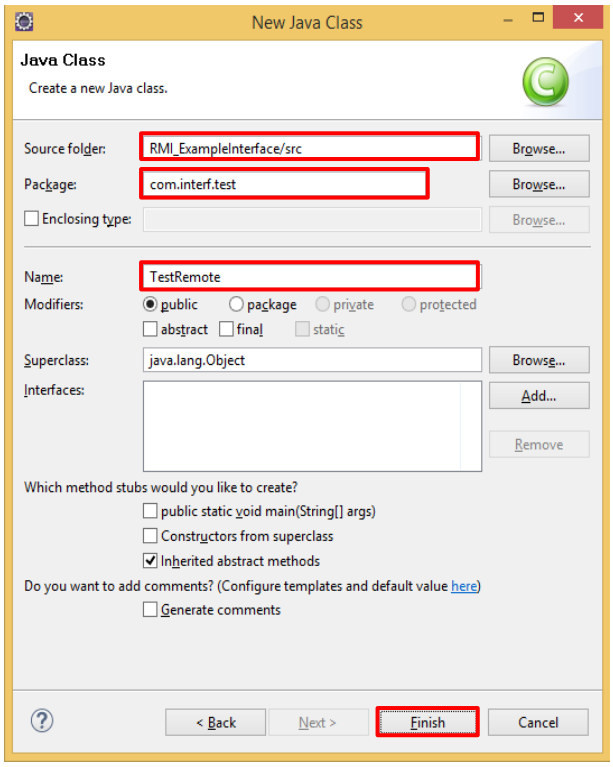
File -> New -> Other -> Java -> Java Project



File -> New -> Other -> Java -> Package

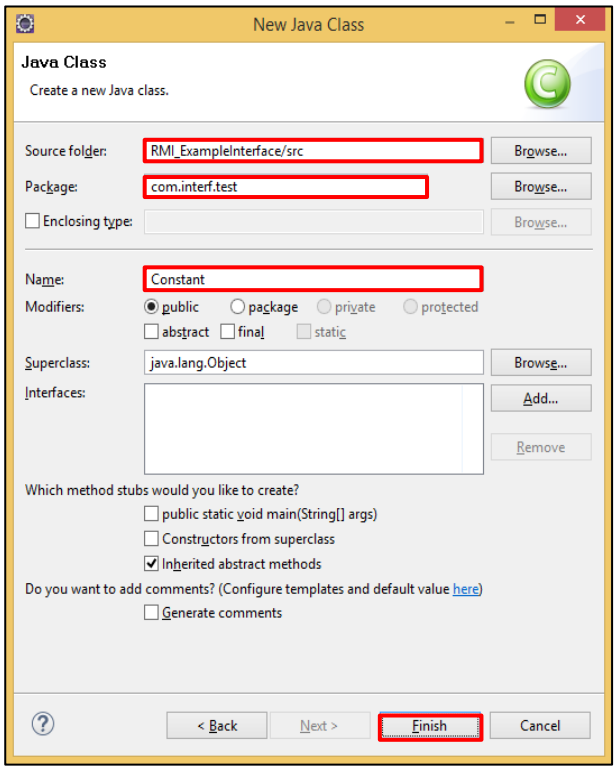


File -> New -> Other -> Java -> Class



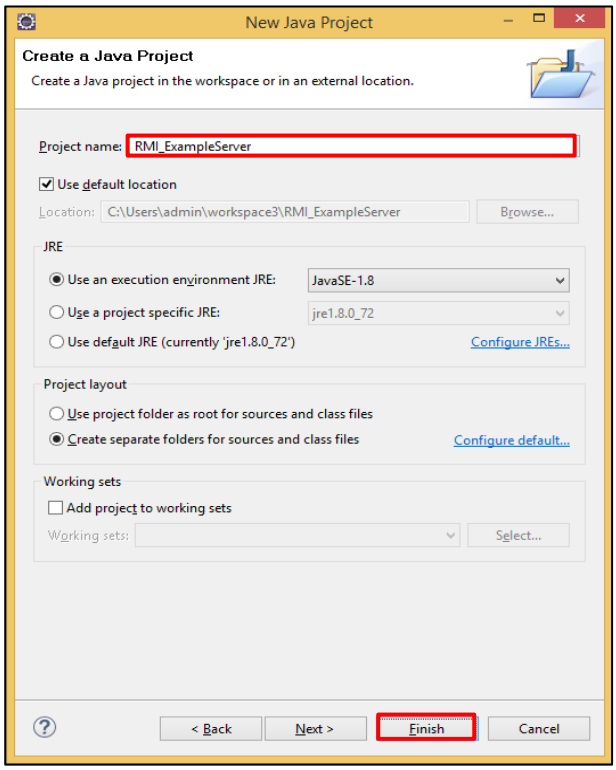

```
package com.interf.test;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface TestRemote extends Remote{
        public boolean isLoginValid(String username) throws RemoteException;
}
```

File -> New -> Other -> Java -> Class

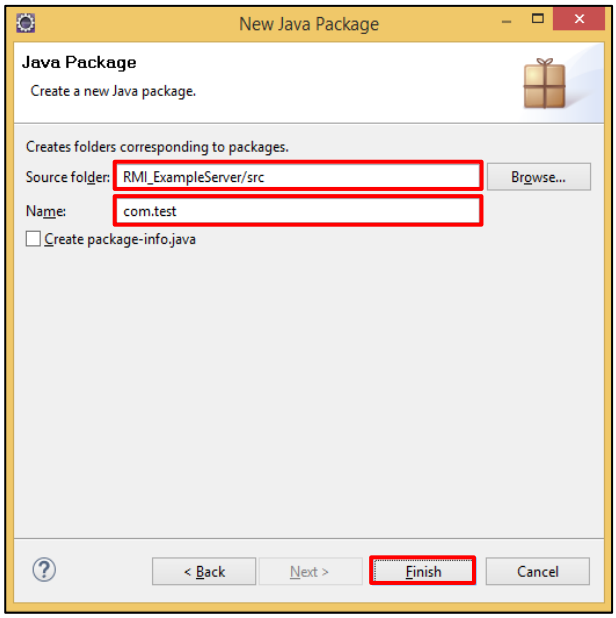



```
package com.interf.test;

public class Constant {
        public static final String RMI_ID = "TestRMI";
        public static final int RMI_PORT = 222;
}
```

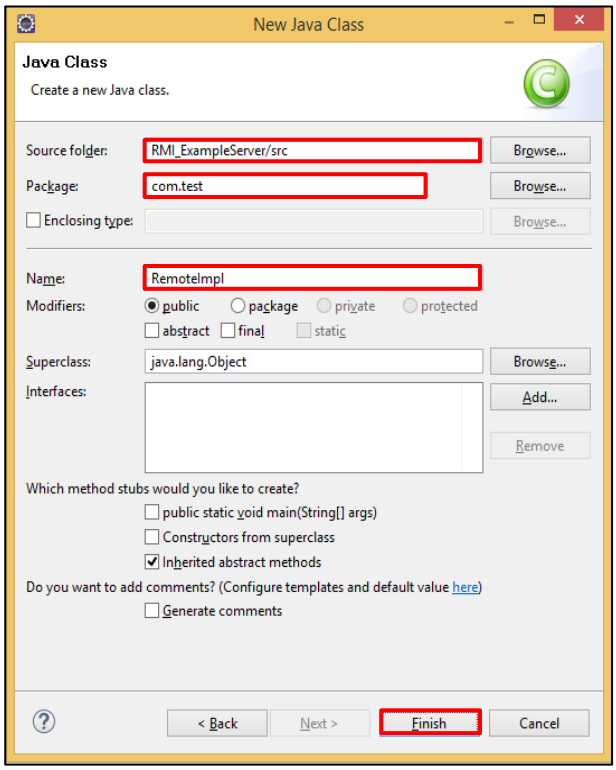
File -> New -> Other -> Java -> Java Project



File -> New -> Other -> Java -> Java Package

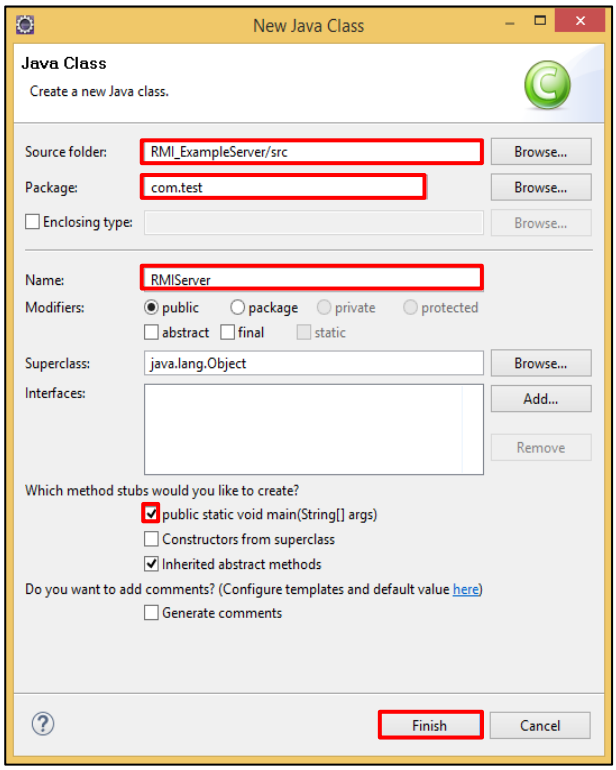


File -> New -> Other -> Java -> Class



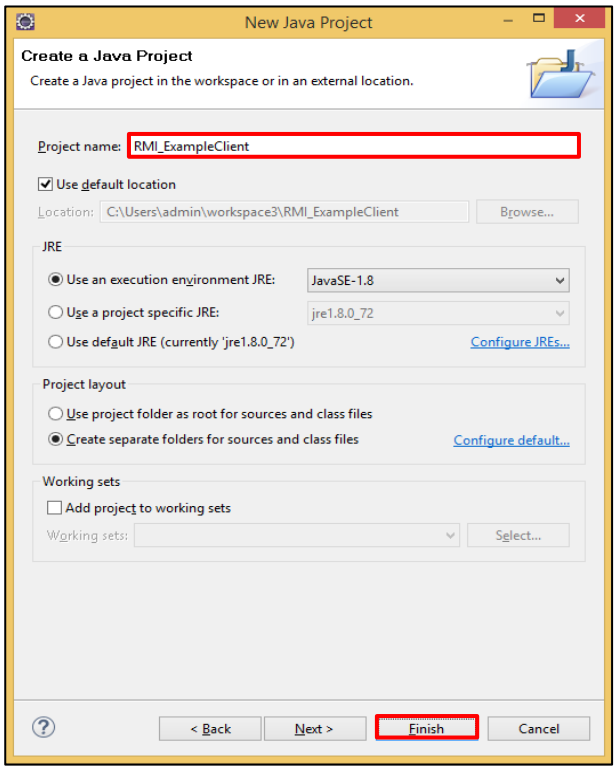

```java
package com.test;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

import com.interf.test.TestRemote;

public class RemoteImpl extends UnicastRemoteObject implements TestRemote {

	private static final long serialVersionUID = 1L;

	protected RemoteImpl() throws RemoteException {
		super();
	}

	@Override
	public boolean isLoginValid(String username) throws RemoteException {
		if (username.equals("test")) {
			return true;
		}
		return false;
	}
}
```

File -> New -> Other -> Java -> Class

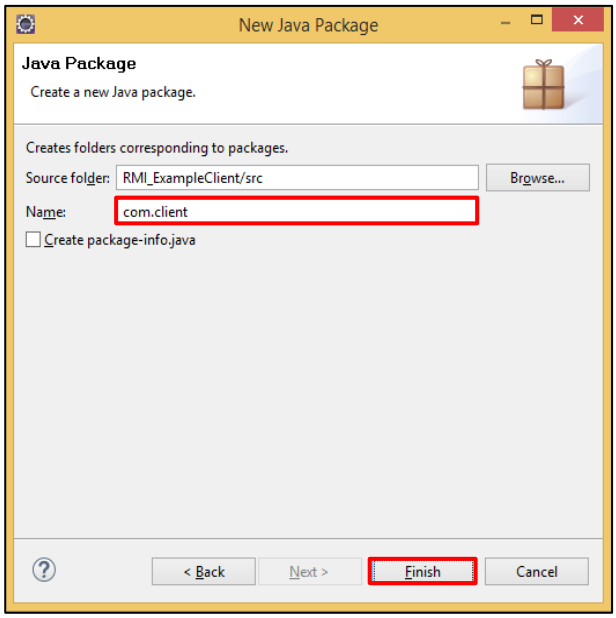



```java
package com.test;

import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

import com.interf.test.Constant;

public class RMIServer {

        public static void main(String[] args)
                        throws RemoteException, AlreadyBoundException {
                RemoteImpl impl = new RemoteImpl();
                Registry registry = LocateRegistry.createRegistry(Constant.RMI_PORT);
                registry.bind(Constant.RMI_ID,impl);
                System.out.println("Server is started!");
        }
}
```

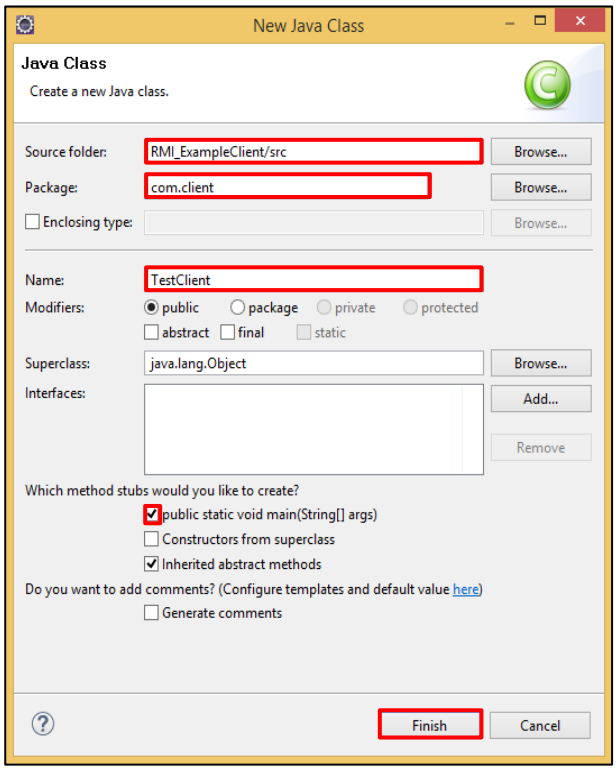
File -> New -> Other -> Java -> Java Project



File -> New -> Other -> Java -> Package



File -> New -> Other -> Java -> Class




```
package com.client;

import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

import com.interf.test.Constant;
import com.interf.test.TestRemote;

public class TestClient {
   public static void main(String[] args)
               throws RemoteException, NotBoundException {
         Registry registry = LocateRegistry.getRegistry("localhost", Constant.RMI_PORT);
         TestRemote remote = (TestRemote) registry.lookup(Constant.RMI_ID);
         System.out.println(remote.isLoginValid("test"));
         System.out.println(remote.isLoginValid("TEST"));
   }
}
```

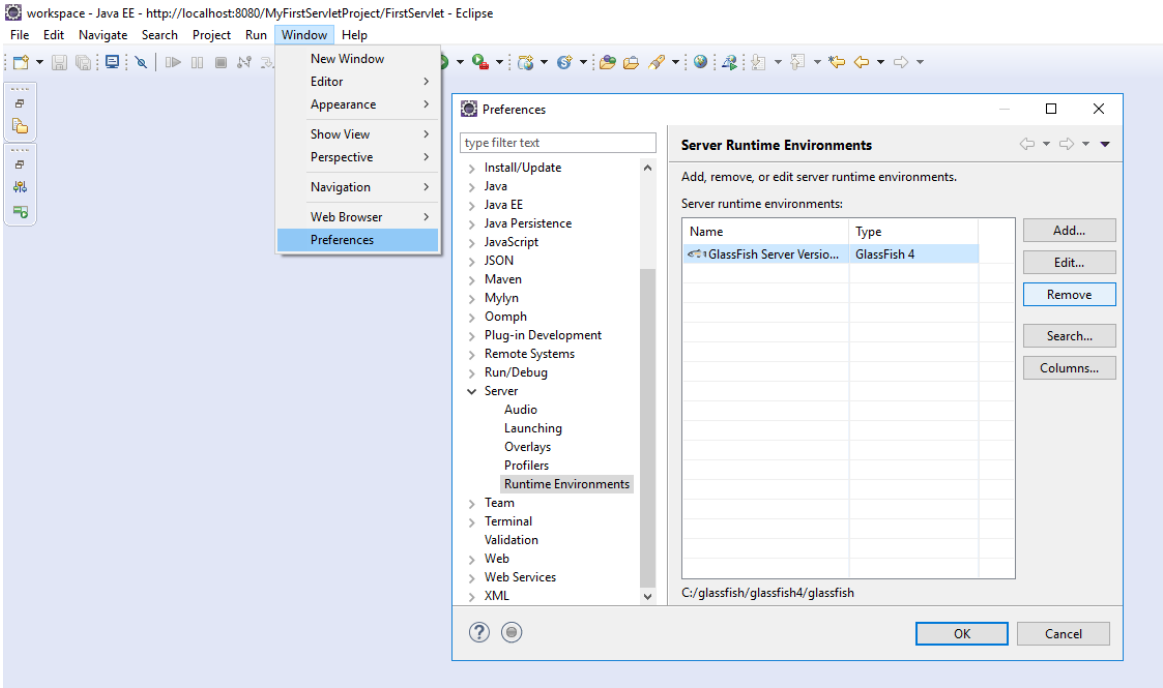
Troubleshooting "DeploymentException" on Glassfish startup

The problem is discussed here.

Whenever you encounter a "`org.glassfish.deployment.common.DeploymentException`" when starting the server, try this:

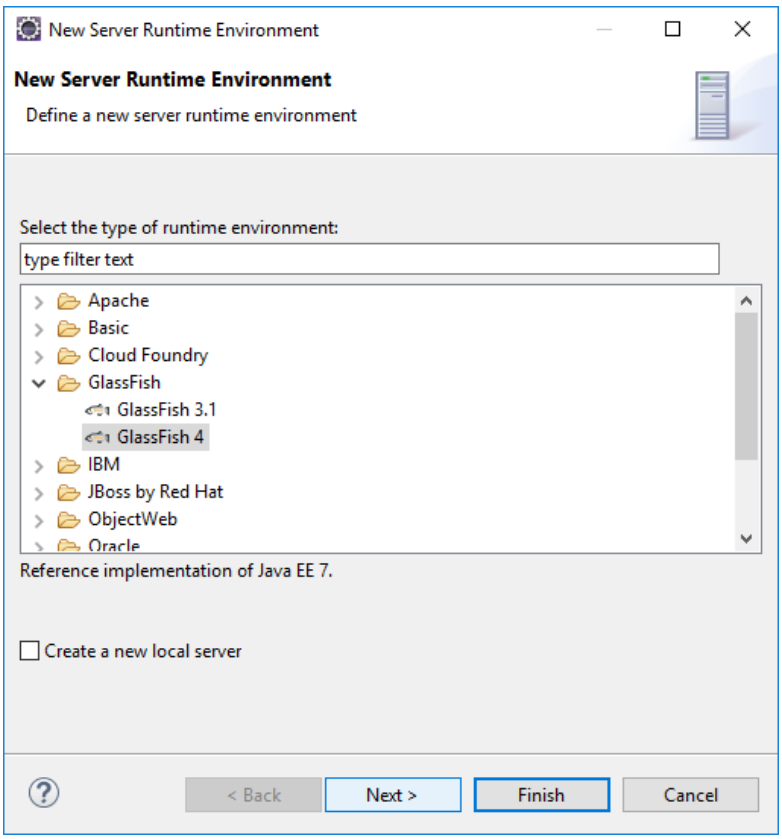
- Stop the server.
- Navigate to the glassfish installation directory and delete the contents of the "generated" folder.
- Try to restart the server.
- If the error persists, follow with the next steps.

- In the Eclipse Menubar, navigate to Window > Preferences > Server > Runtime Environments and select the Glassfish server and delete it:



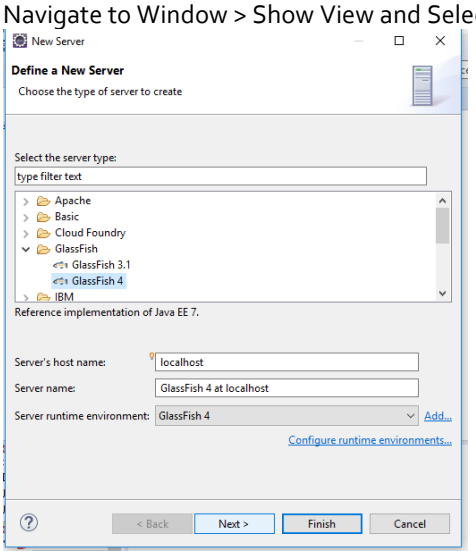
Then close eclipse and proceed with the next step.

- Open an Explorer window and navigate to you Glassfish installation folder. Then navigate to /domains/ and delete every file and folder in there.
- Now you can open up eclipse again and navigate to Window > Preferences > Server > Runtime Environments like before. There you click on Add...



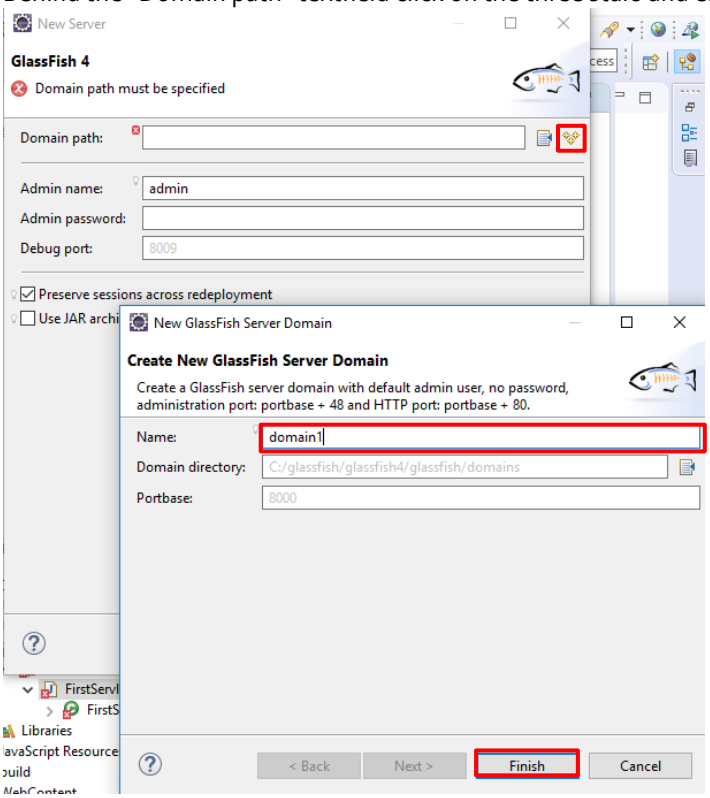
Leave the "Create a new local server"-Checkbox unticked! Then click on Next.

- Enter a Name for the server (doesn't matter) and the path to the glassfish installation in the appropriate box. Then click on finish
- Navigate to Window > Show View and Select "Servers" then click on the blue, underlined text.



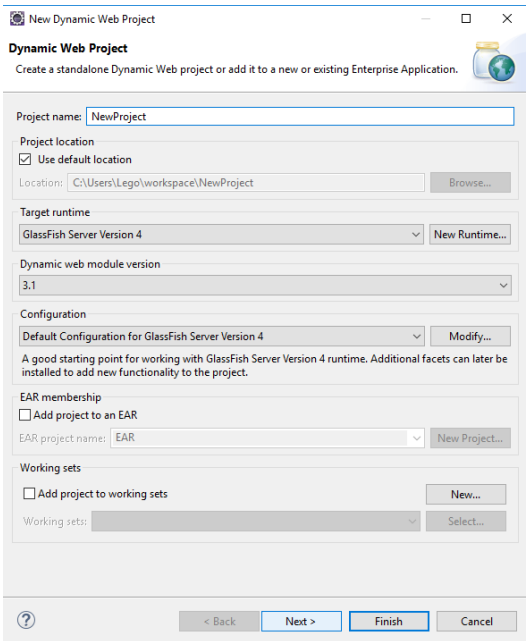
Then click on Next.

- Behind the "Domain path" textfield click on the three stars and enter a domainname



Then click on finish and in the other window also on Finish

- Now we have to create a new Eclipse project and copy all files and folders from the old project to the new one. This includes all java source files, webfiles like .html .xhtml .jsf.jsp. This is accomplished by following the next steps.
- Navigate to File > New > Dynamic Web Project. Target Runtime should be the Glassifsh runtime created before.



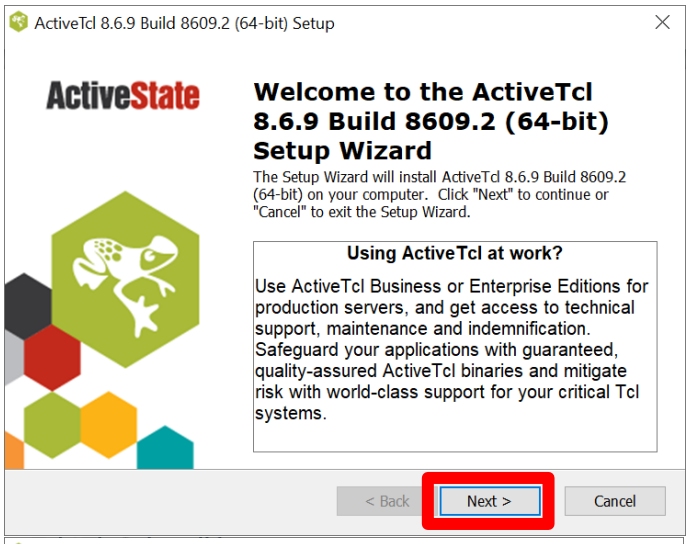
Click on Finish

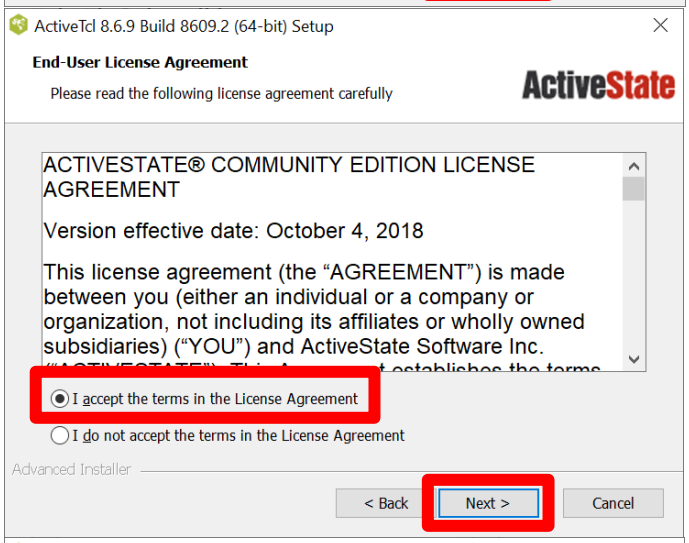
- Copy all webfiles from WebContent to the new project into the same folder. The web.xml should also be copied in the WEB-INF folder. The file glassfish-web.xml shouldn't be copied but can be editet to match additional configuration from the original Project. All Java Sources should be copied into the new project with the same package names.
- In you newly created project you shouldn't see any errors related to not finding Servlet classes. This means everything is properly set up and you can launch your application.

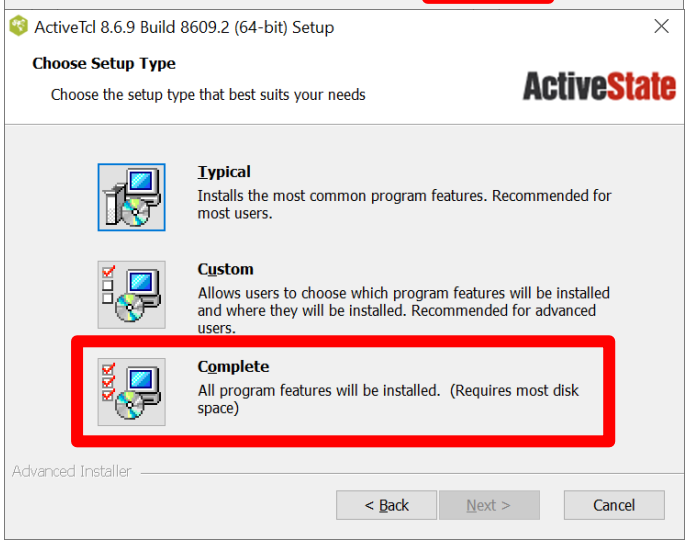
Tcl / Tk Gui toolkit

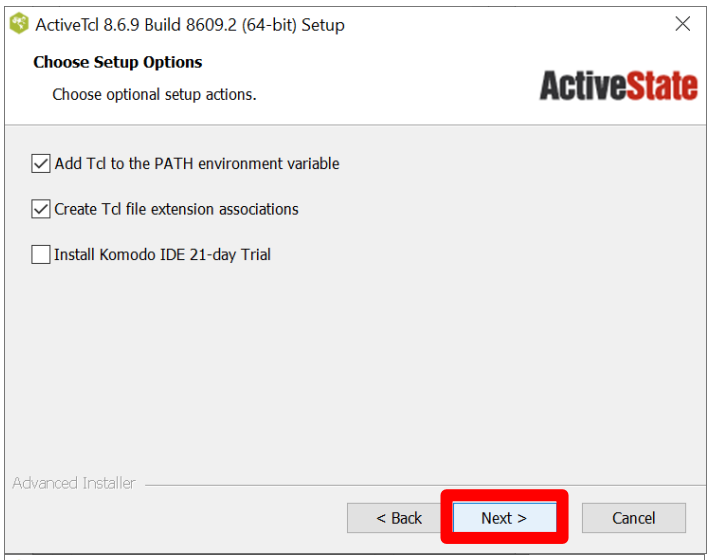
Active Tcl von Active State

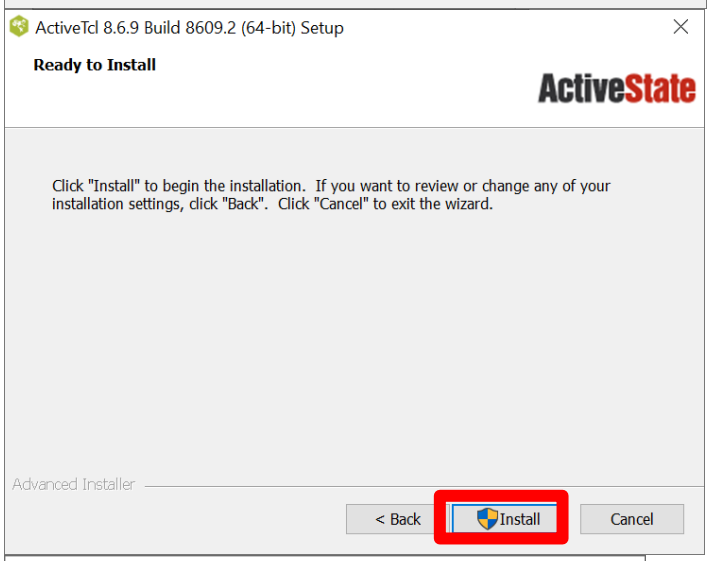
Download https://www.activestate.com/products/tcl/downloads/ (use your GitHup Account)

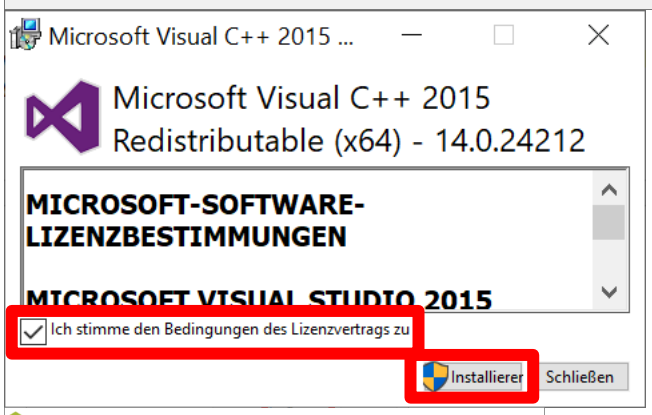







ActiveTcl 8.6.9 Build 8609.2 (64-bit) Setup
Choose Setup Options
Choose optional setup actions.
ActiveState
Add Tcl to the PATH environment variable
Create Tcl file extension associations
Install Komodo IDE 21-day Trial
Advanced Installer
< Back
Next >
Cancel


ActiveTcl 8.6.9 Build 8609.2 (64-bit) Setup
Ready to Install
ActiveState
Click "Install" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.
Advanced Installer
< Back
Install
Cancel

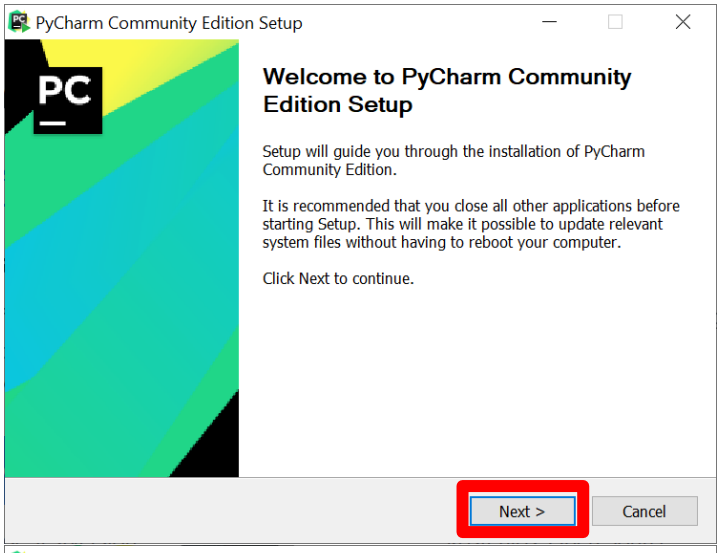

Microsoft Visual C++ 2015 ...
Microsoft Visual C++ 2015
Redistributable (x64) - 14.0.24212
MICROSOFT-SOFTWARE-
LIZENZBESTIMMUNGEN
MICROSOFT VISUAL STUDIO 2015
Ich stimme den Bedingungen des Lizenzvertrags zu
Installieren
Schließen

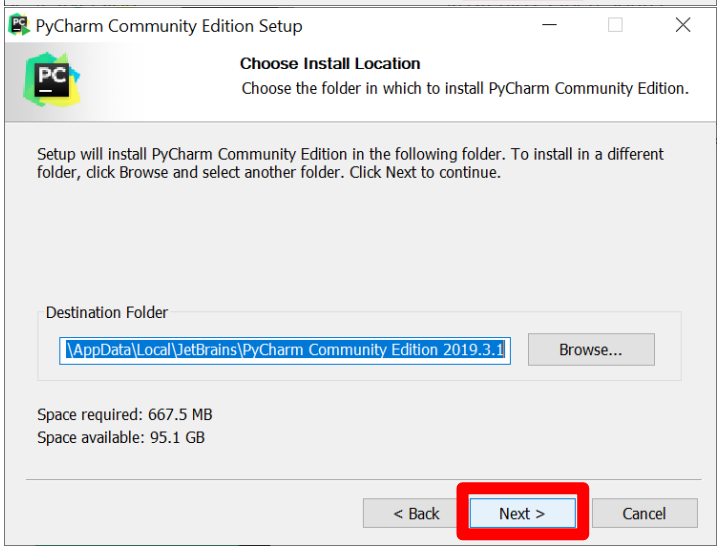
ActiveTcl 8.6.9 Build 8609.2 (64-bit) Setup
ActiveState
Completing the ActiveTcl 8.6.9 Build 8609.2 (64-bit) Setup Wizard
Click the "Finish" button to exit the Setup Wizard.
Finish
Cancel

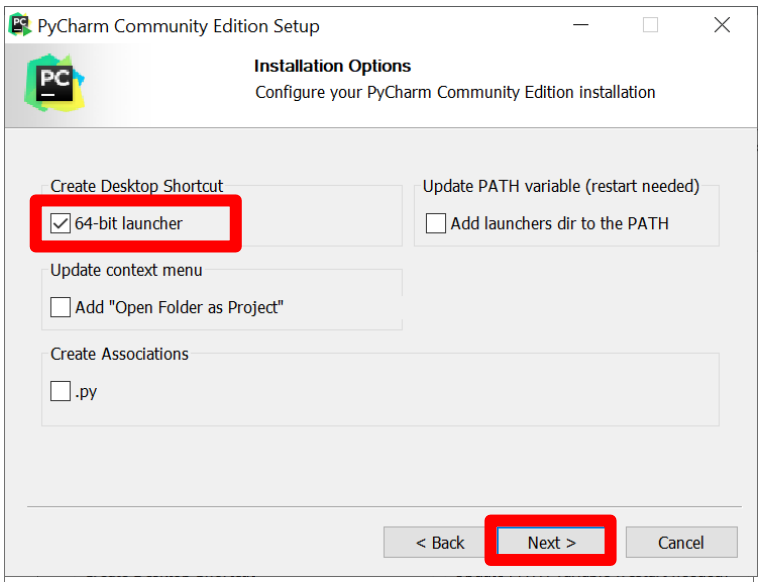
Python

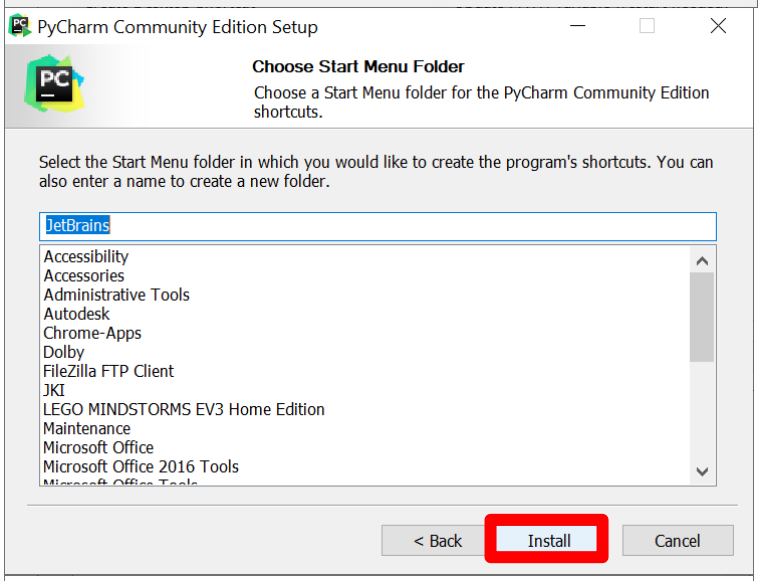
PyCharm von JetBrains

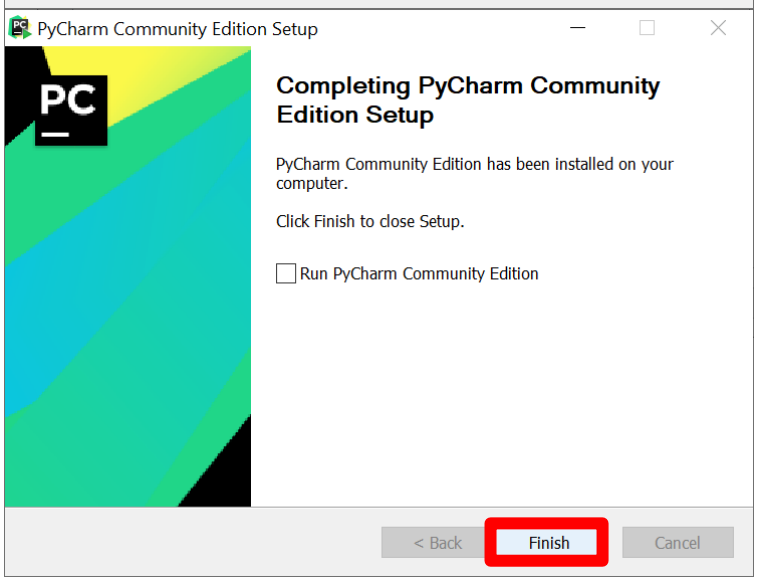
Download free Community Version:
https://www.jetbrains.com/de-de/pycharm/download/#section=windows





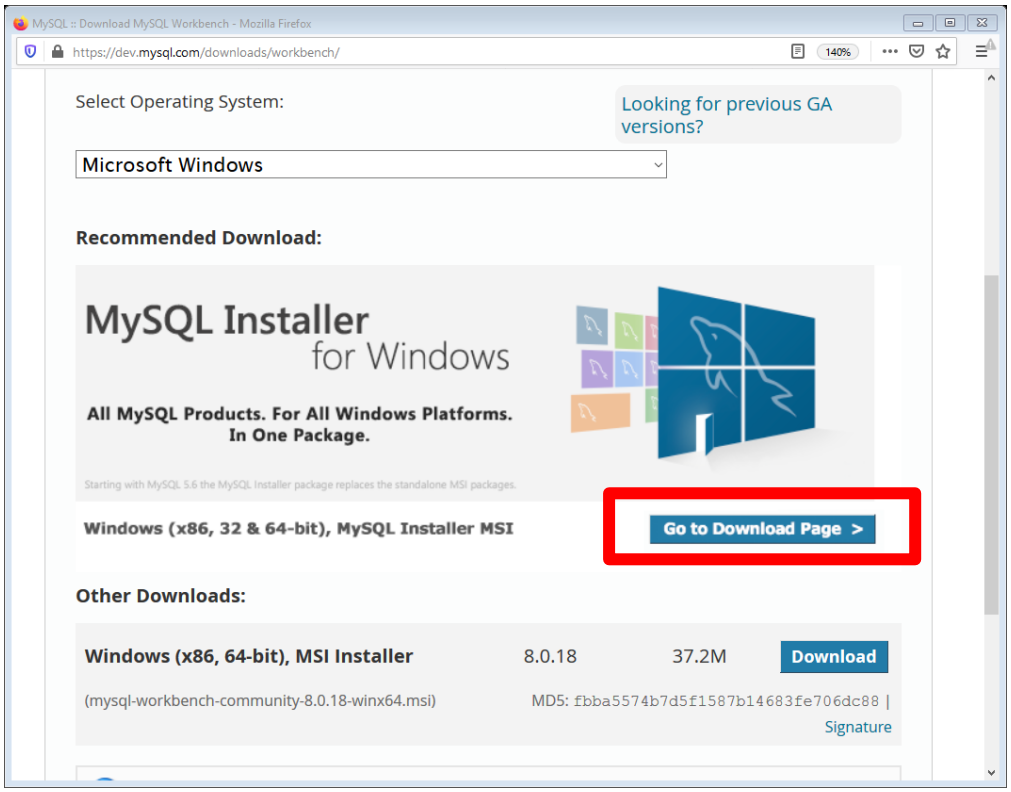

PyCharm Community Edition Setup
Installation Options
Configure your PyCharm Community Edition installation
Create Desktop Shortcut
64-bit launcher
Update PATH variable (restart needed)
Add launchers dir to the PATH
Update context menu
Add "Open Folder as Project"
Create Associations
.py
< Back
Next >
Cancel

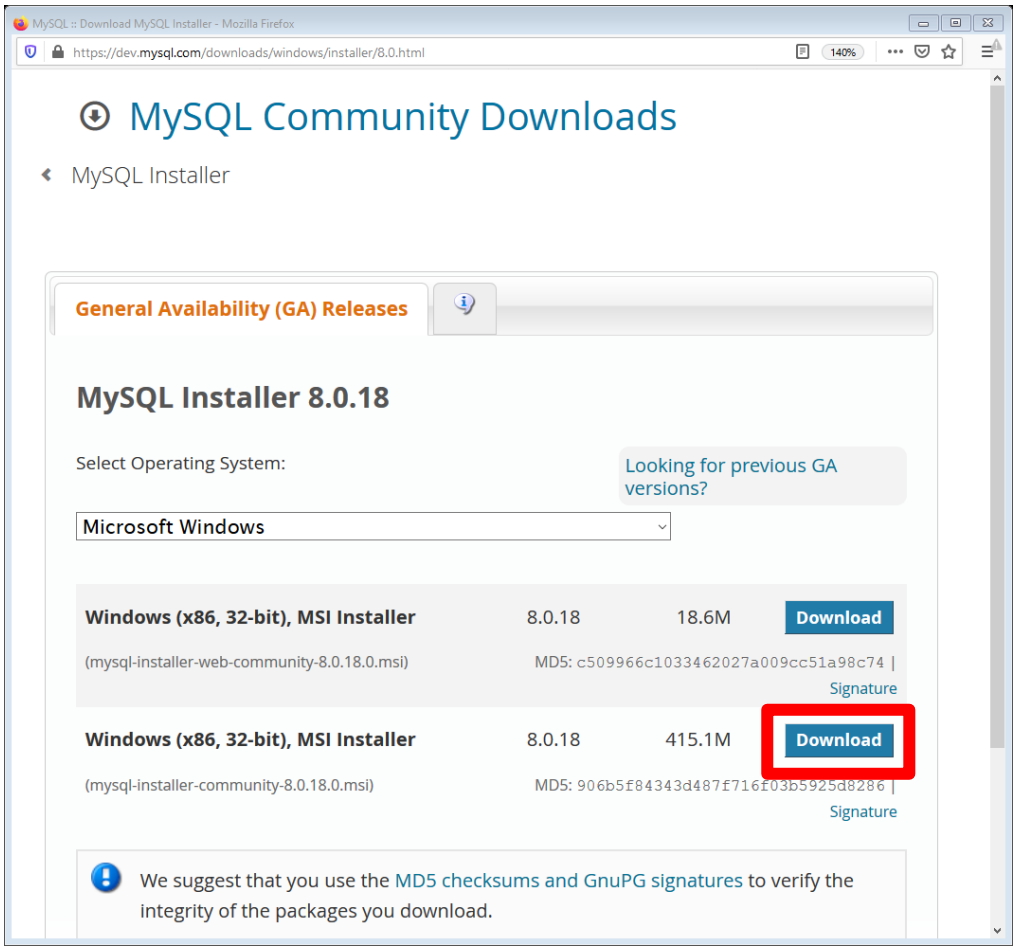

PyCharm Community Edition Setup
Choose Start Menu Folder
Choose a Start Menu folder for the PyCharm Community Edition shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
JetBrains
Accessibility
Accessories
Administrative Tools
Autodesk
Chrome-Apps
Dolby
FileZilla FTP Client
JKI
LEGO MINDSTORMS EV3 Home Edition
Maintenance
Microsoft Office
Microsoft Office 2016 Tools
< Back
Install
Cancel

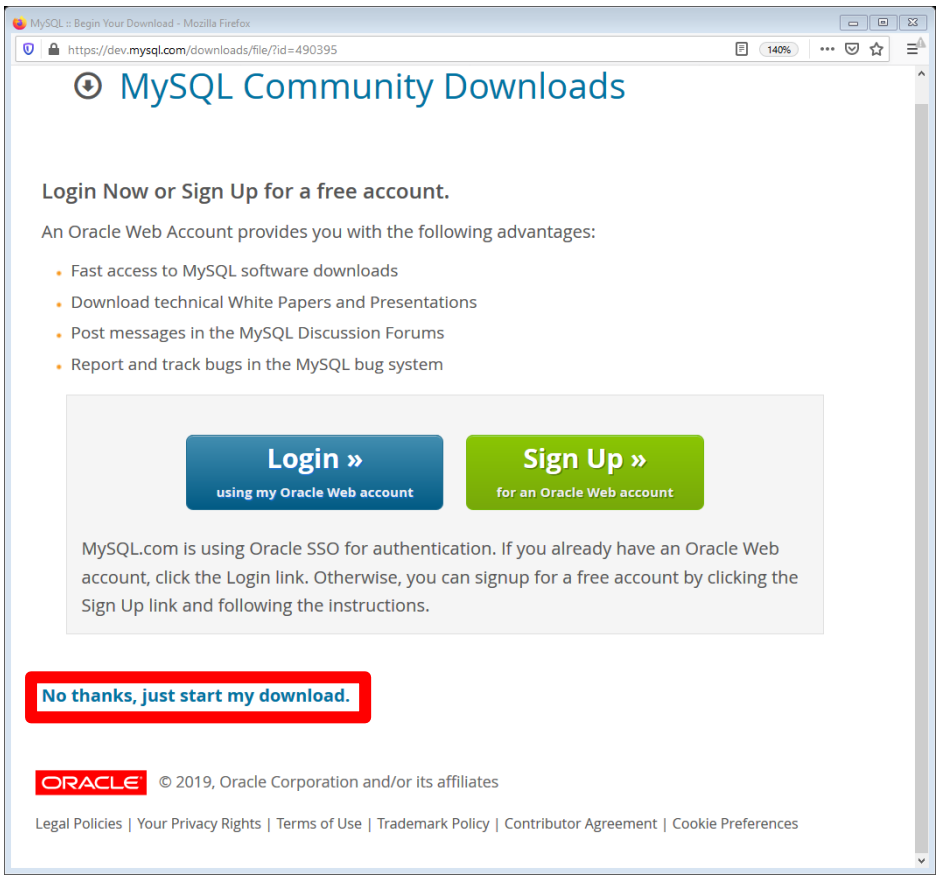

PyCharm Community Edition Setup
Completing PyCharm Community Edition Setup
PyCharm Community Edition has been installed on your computer.
Click Finish to close Setup.
Run PyCharm Community Edition
< Back
Finish
Cancel

MySql (Workbench 8.0.18)

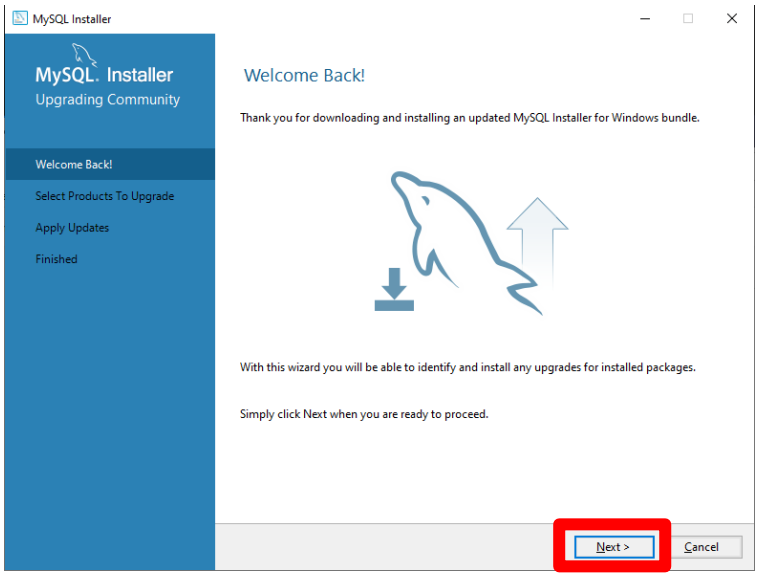
https://dev.mysql.com/downloads/workbench/

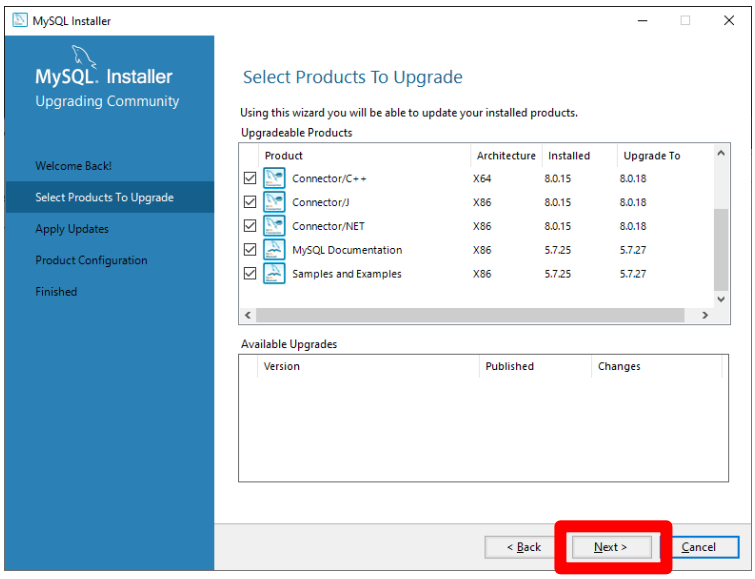


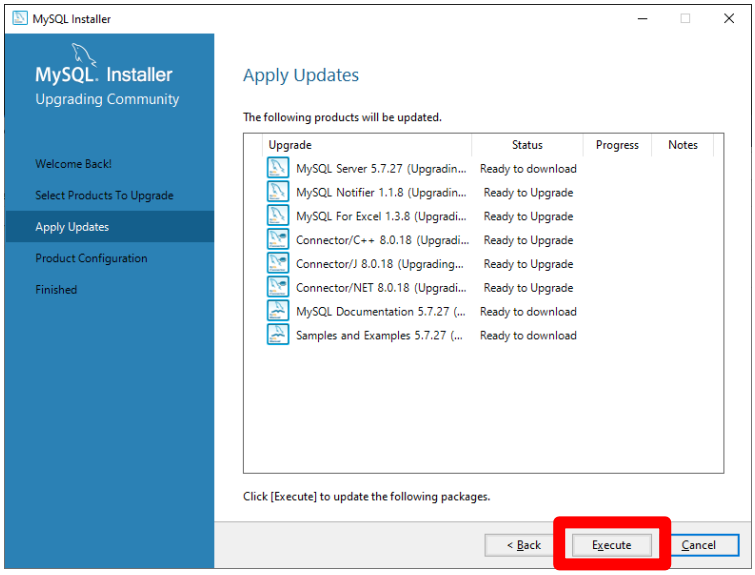




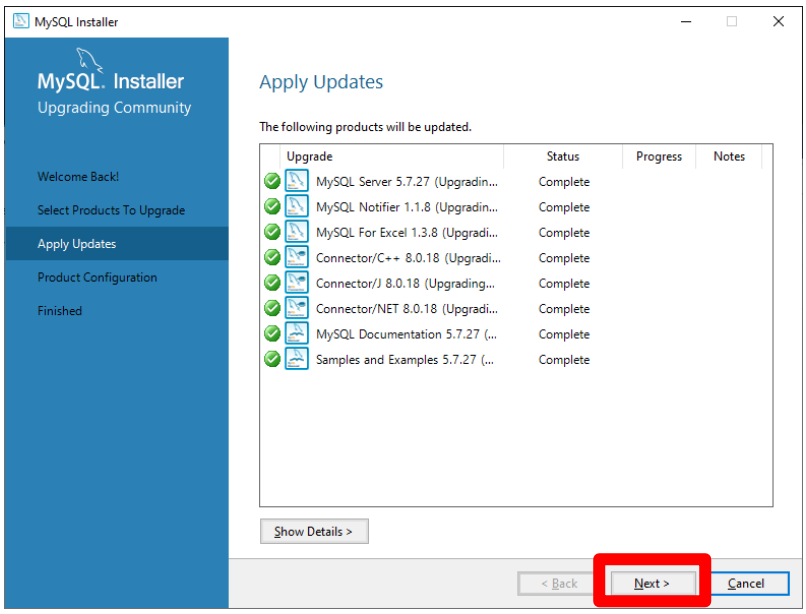
Datei downloaden, speichern und Installation starten.



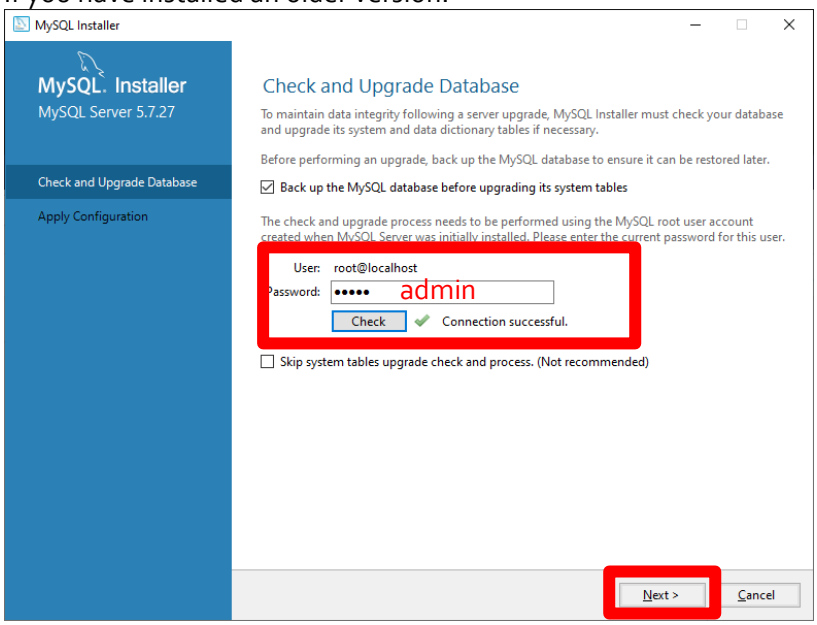

MySQL Installer
MySQL. Installer
Upgrading Community
Welcome Back!
Select Products To Upgrade
Apply Updates
Product Configuration
Finished
Select Products To Upgrade
Using this wizard you will be able to update your installed products.
Upgradeable Products
Product
Architecture
Installed
Upgrade To
Connector/C++ X64 8.0.15 8.0.18
Connector/J X86 8.0.15 8.0.18
Connector/NET X86 8.0.15 8.0.18
MySQL Documentation X86 5.7.25 5.7.27
Samples and Examples X86 5.7.25 5.7.27
Available Upgrades
Version
Published
Changes
< Back
Next >
Cancel

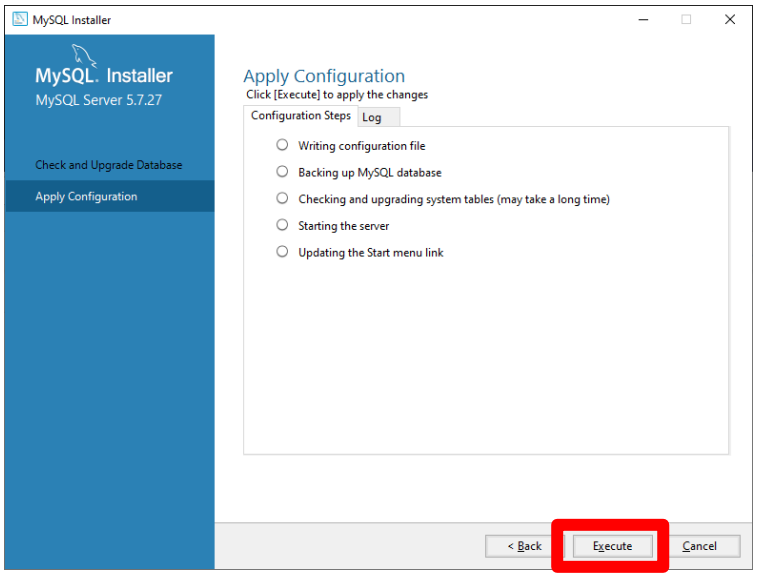

MySQL Installer
MySQL. Installer
Upgrading Community
Welcome Back!
Select Products To Upgrade
Apply Updates
Product Configuration
Finished
Apply Updates
The following products will be updated.
Upgrade
Status
Progress
Notes
MySQL Server 5.7.27 (Upgradin... Ready to download
MySQL Notifier 1.1.8 (Upgradin... Ready to Upgrade
MySQL For Excel 1.3.8 (Upgradi... Ready to Upgrade
Connector/C++ 8.0.18 (Upgradi... Ready to Upgrade
Connector/J 8.0.18 (Upgrading... Ready to Upgrade
Connector/NET 8.0.18 (Upgradi... Ready to Upgrade
MySQL Documentation 5.7.27 (... Ready to download
Samples and Examples 5.7.27 (... Ready to download
Click [Execute] to update the following packages.
< Back
Execute
Cancel

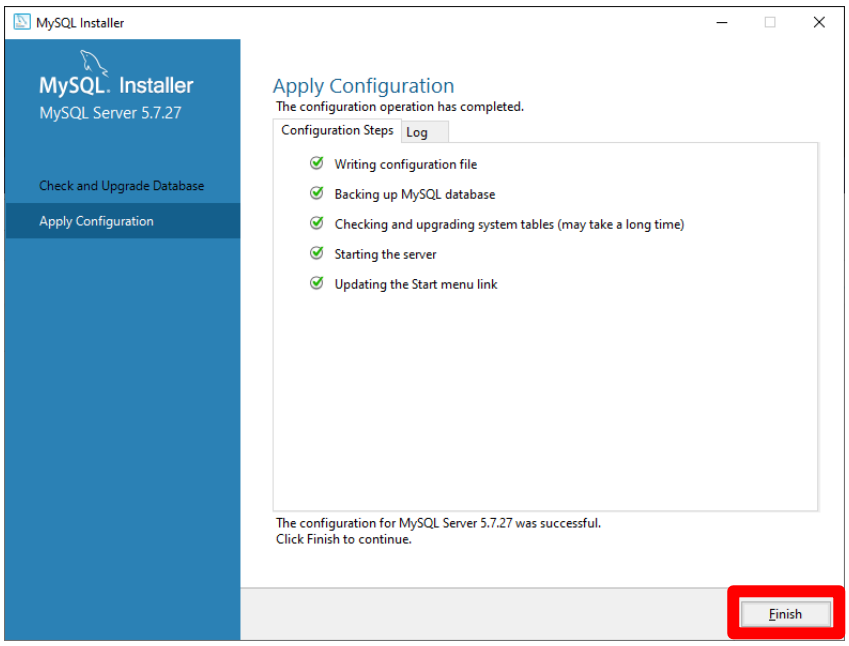
After a while.....



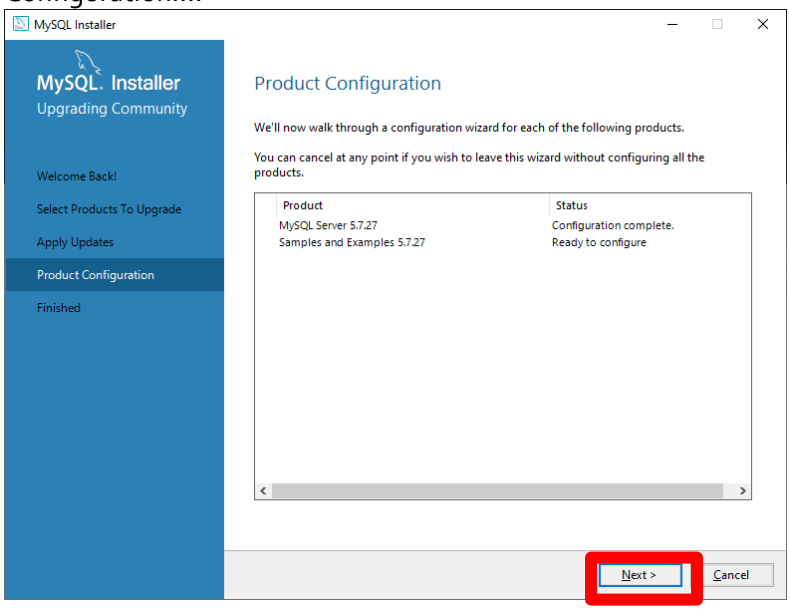
If you have installed an older version:

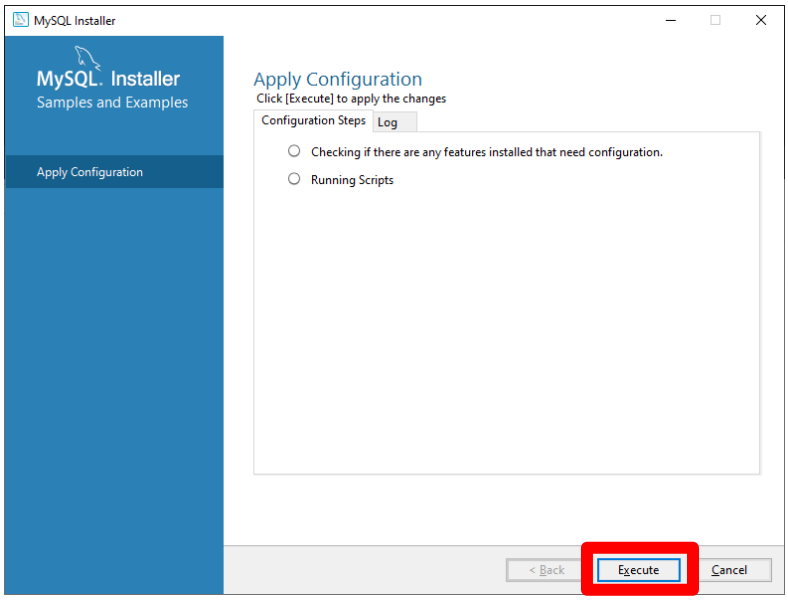


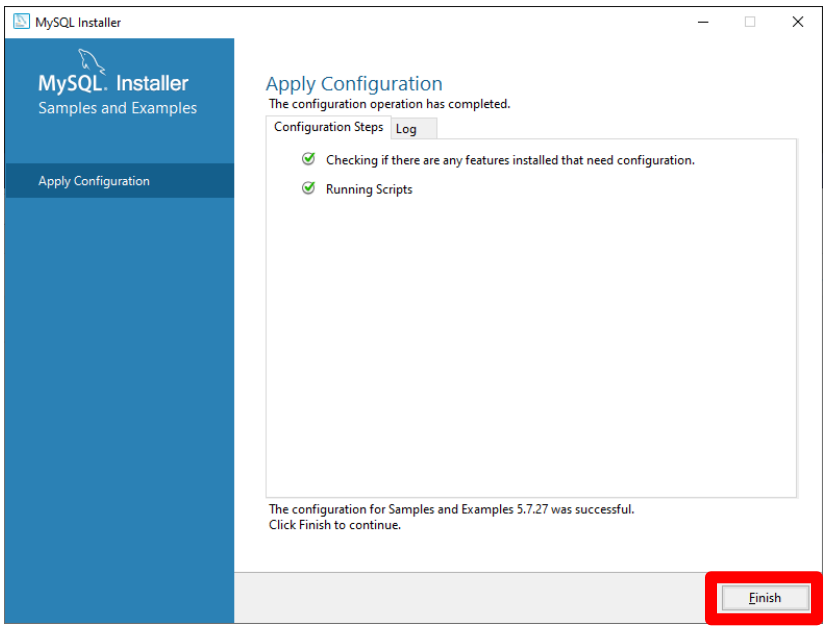


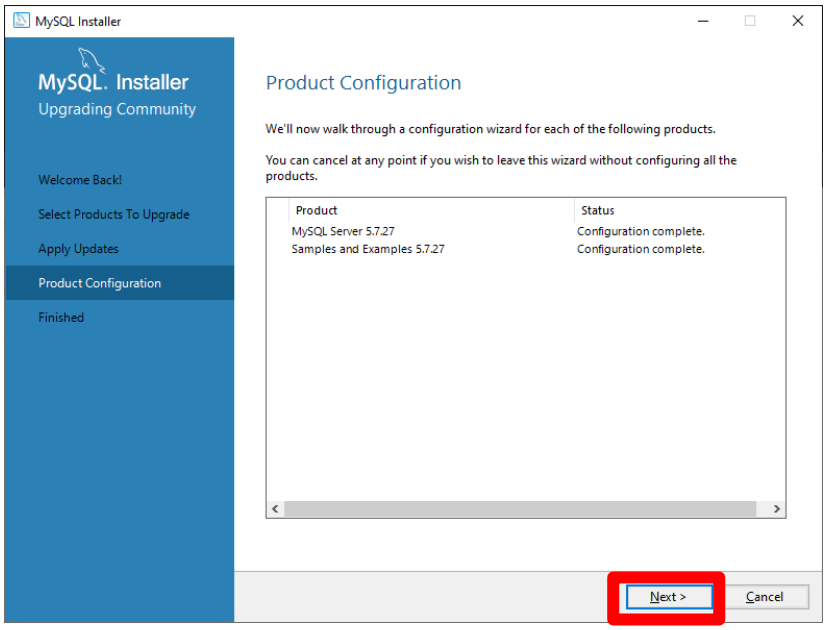


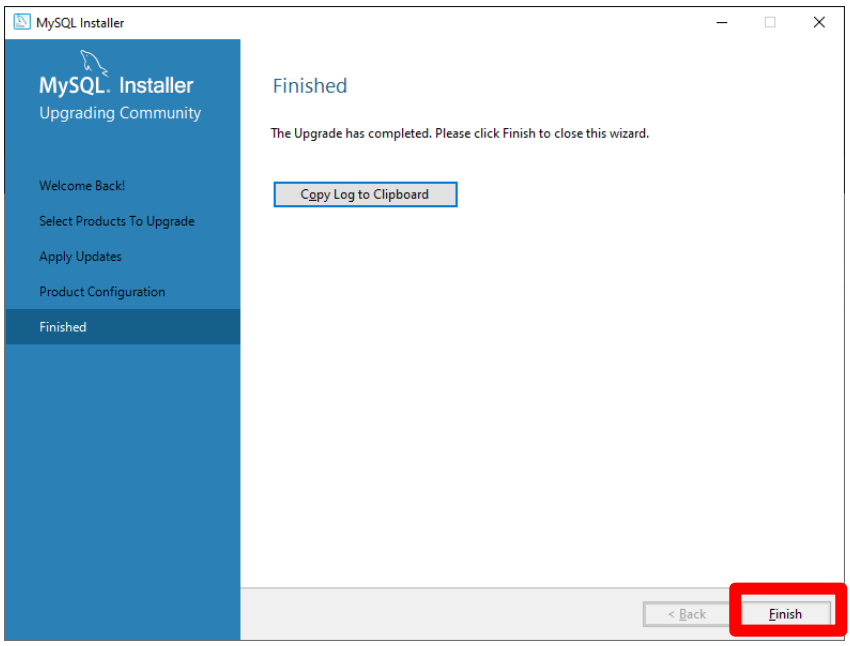
Configuration....



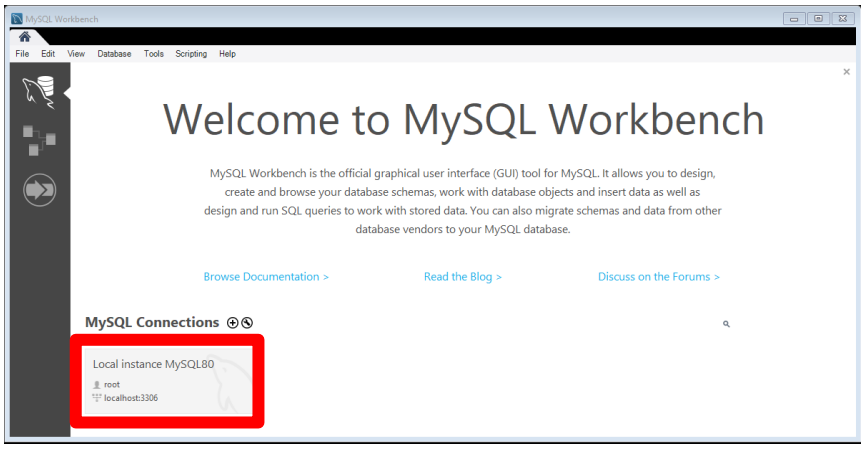


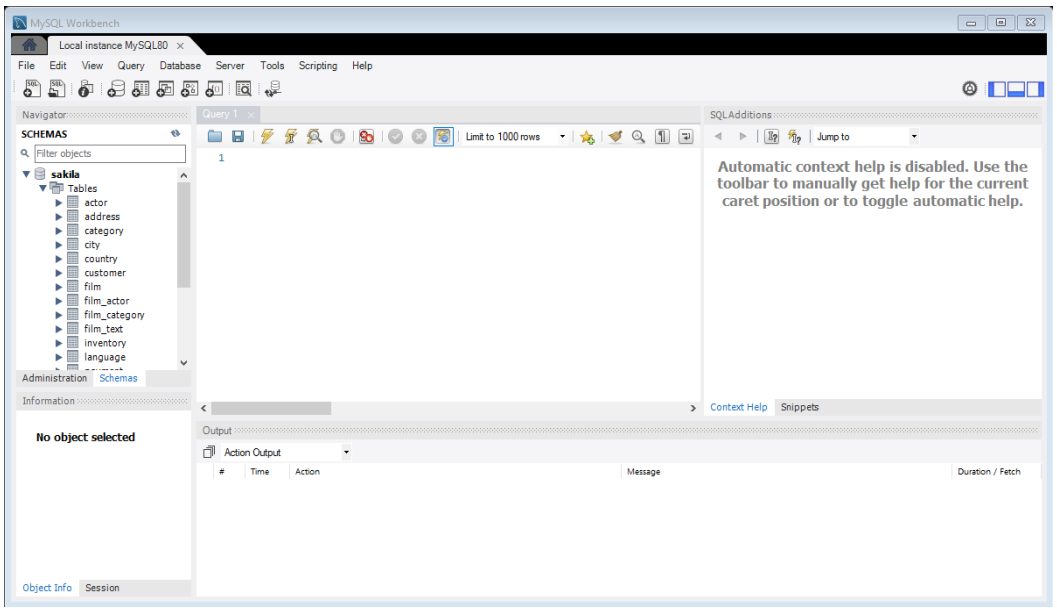

MySQL Installer
MySQL. Installer
Samples and Examples
Apply Configuration
Apply Configuration
The configuration operation has completed.
Configuration Steps
Log
Checking if there are any features installed that need configuration.
Running Scripts
The configuration for Samples and Examples 5.7.27 was successful.
Click Finish to continue.
Finish


MySQL Installer
MySQL. Installer
Upgrading Community
Welcome Back!
Select Products To Upgrade
Apply Updates
Product Configuration
Finished
Product Configuration
We'll now walk through a configuration wizard for each of the following products.
You can cancel at any point if you wish to leave this wizard without configuring all the products.
Product
Status
MySQL Server 5.7.27
Configuration complete.
Samples and Examples 5.7.27
Configuration complete.
Next >
Cancel



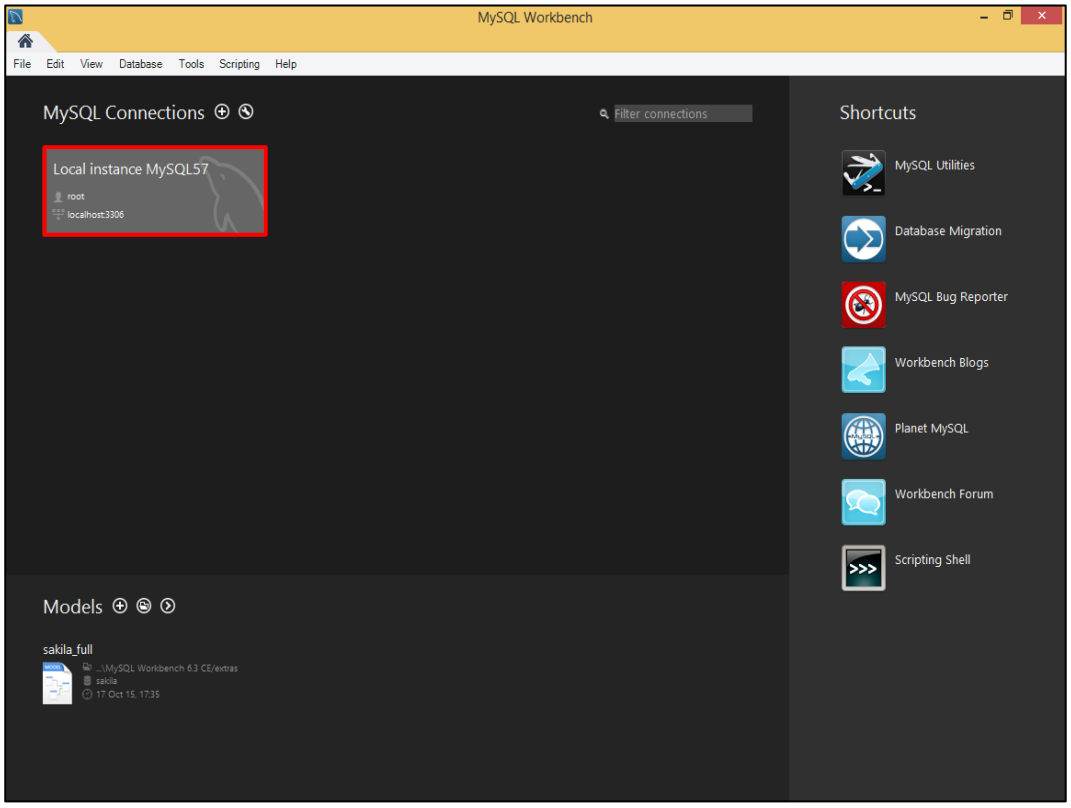
Start workbench



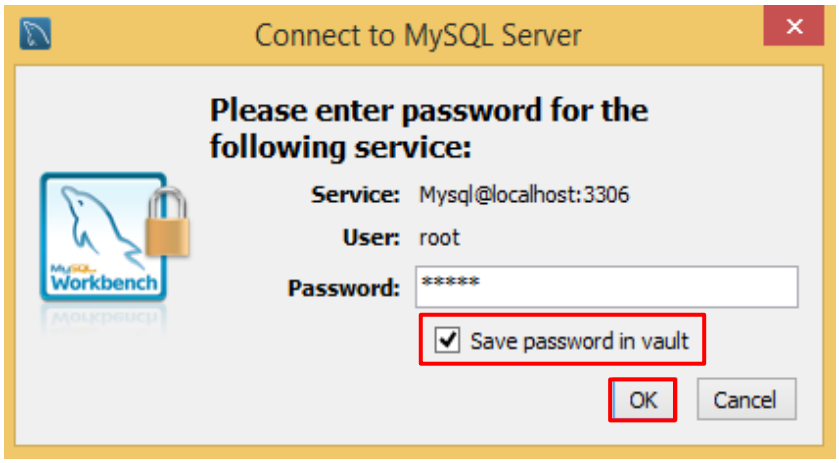


Create local User in MySQL Workbench

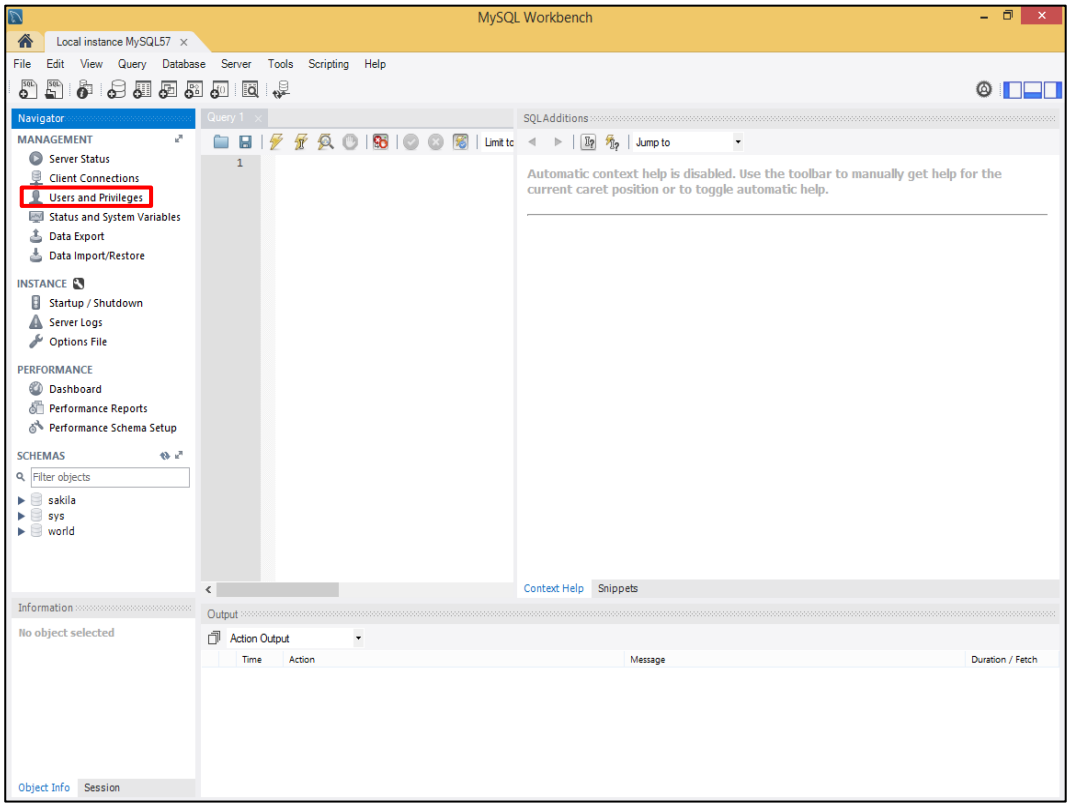
Start -> MySQL Workbench

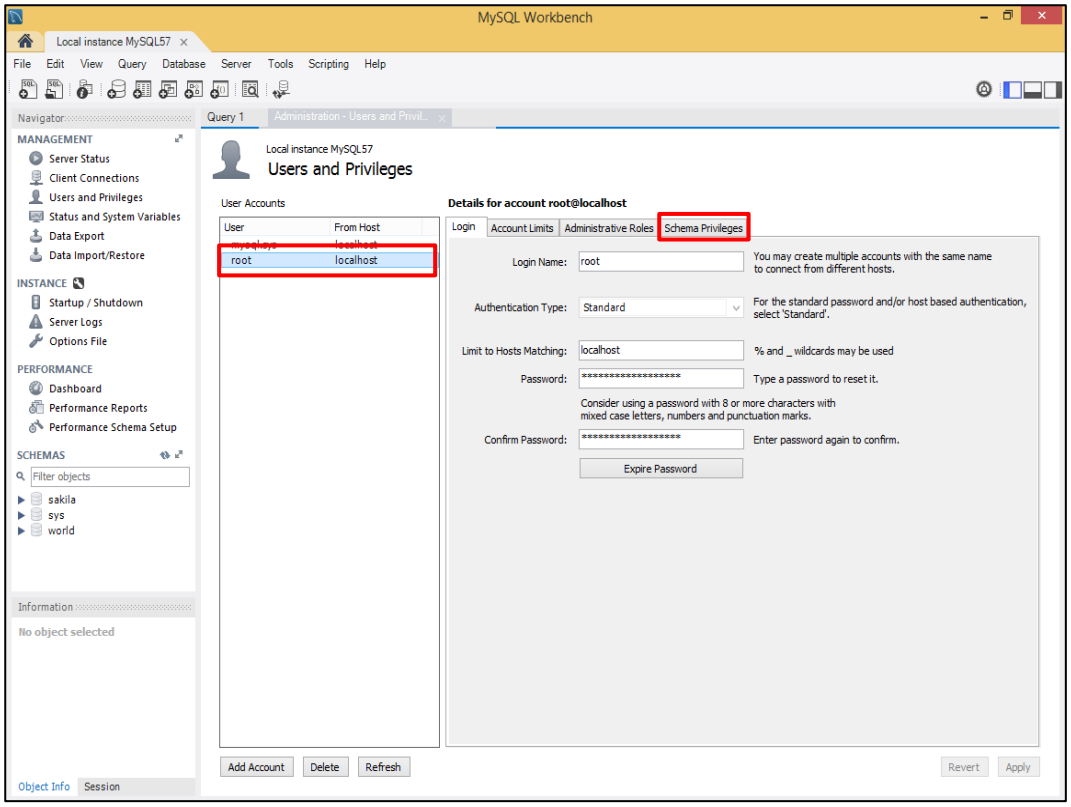


MySQL Root Password: admin

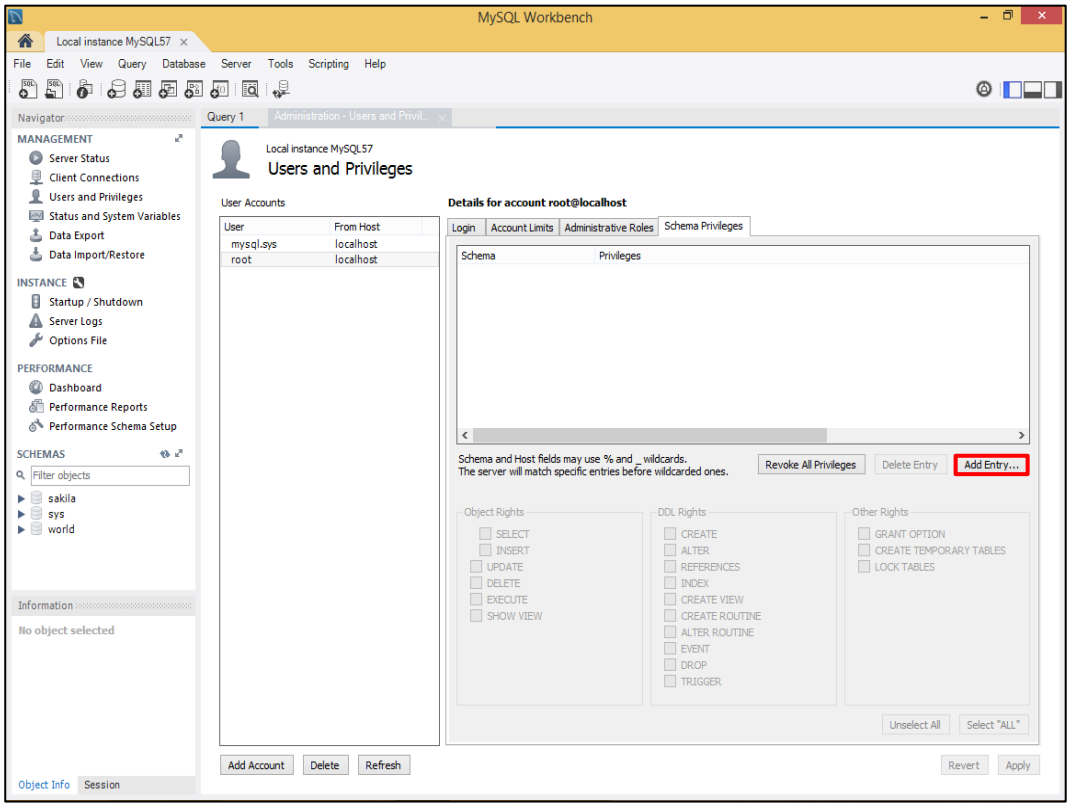


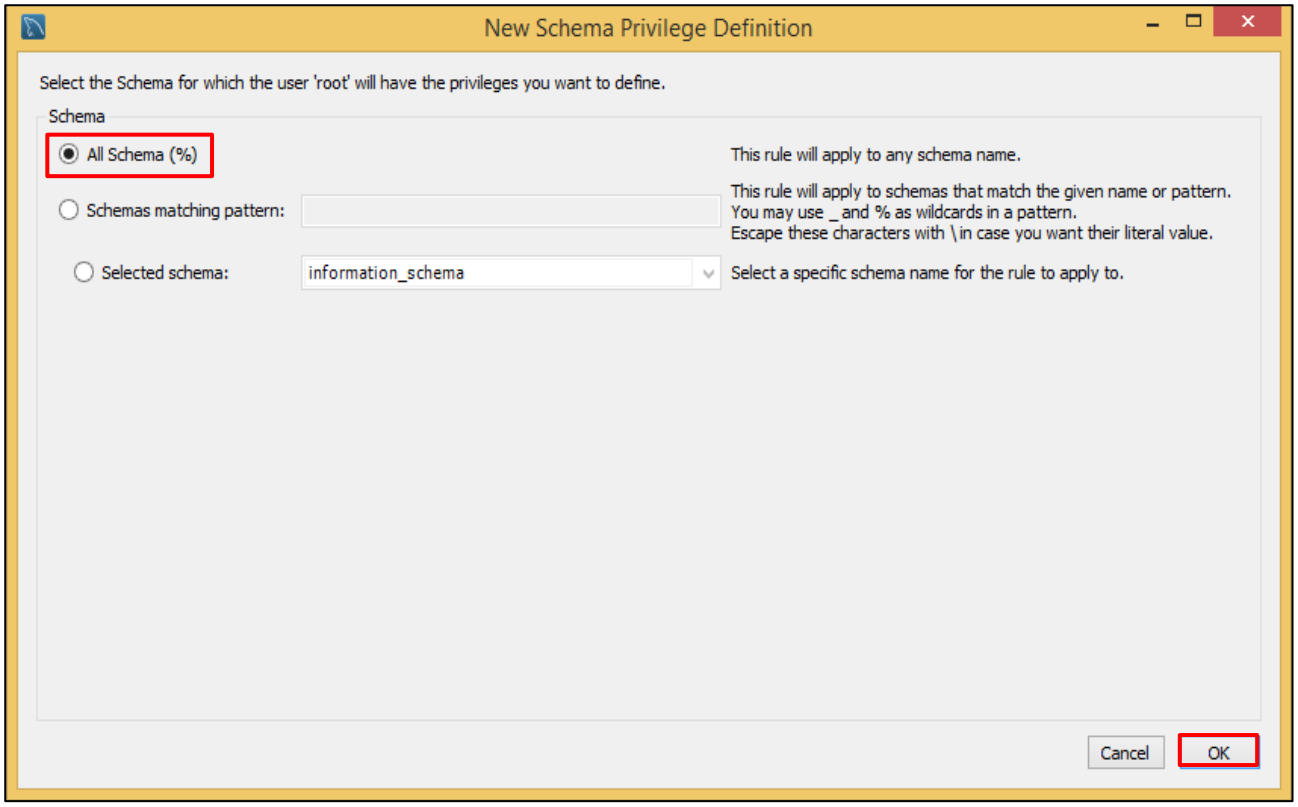
Go to User and Privileges, select root and choose Schema Privileges.



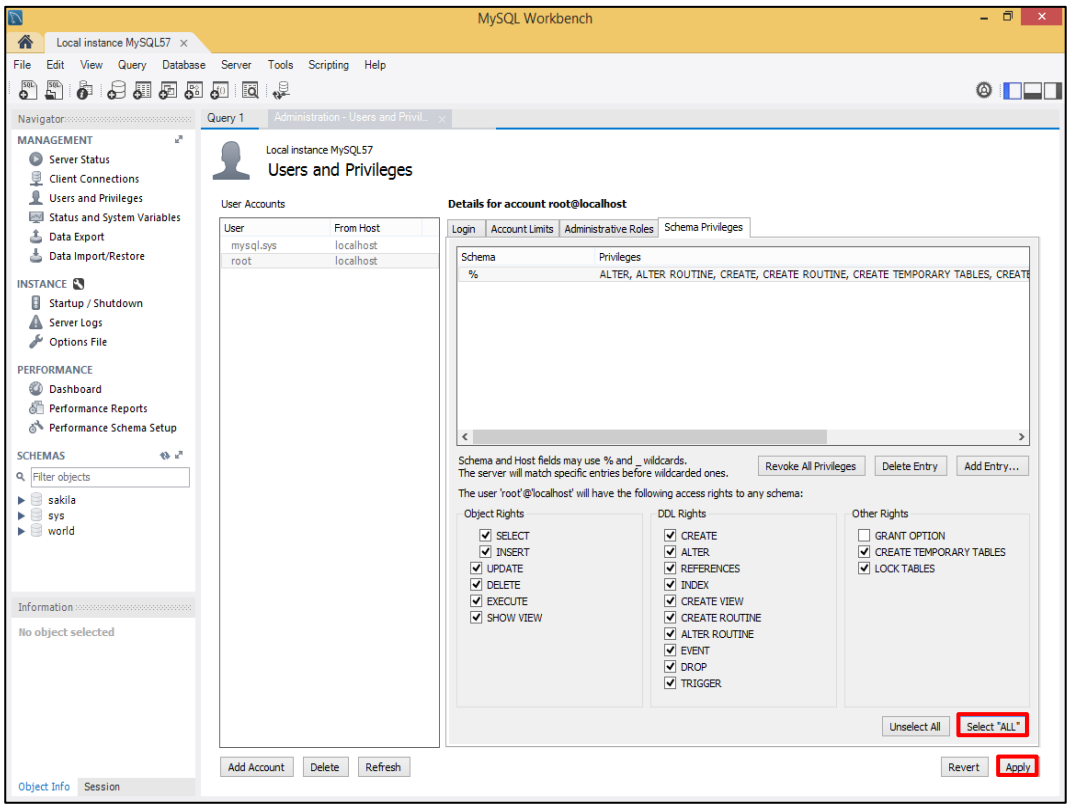


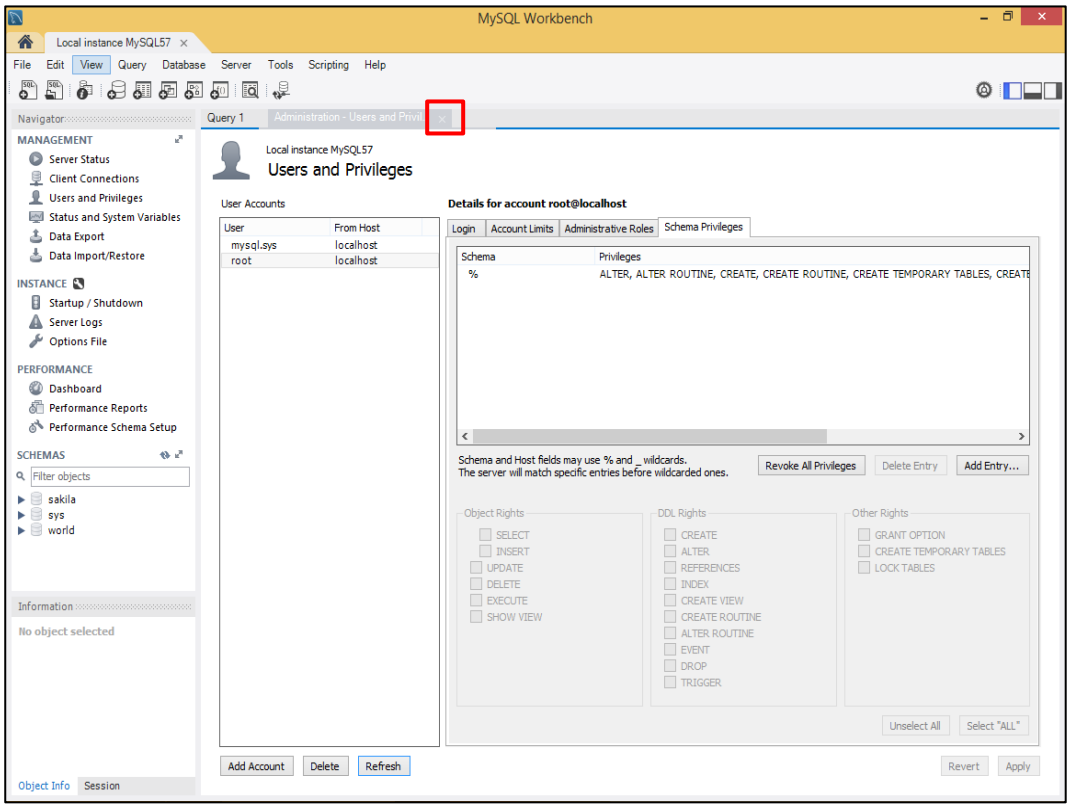
Click on Add Entry button and select All Schema (%).





When you have set the Schema Privilege Definition please click on the following page on Select "All" button and apply it.

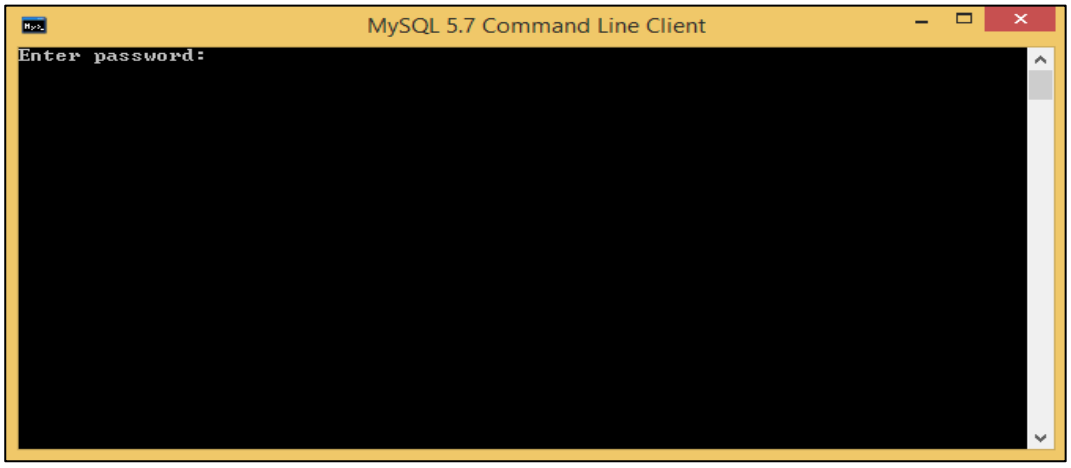


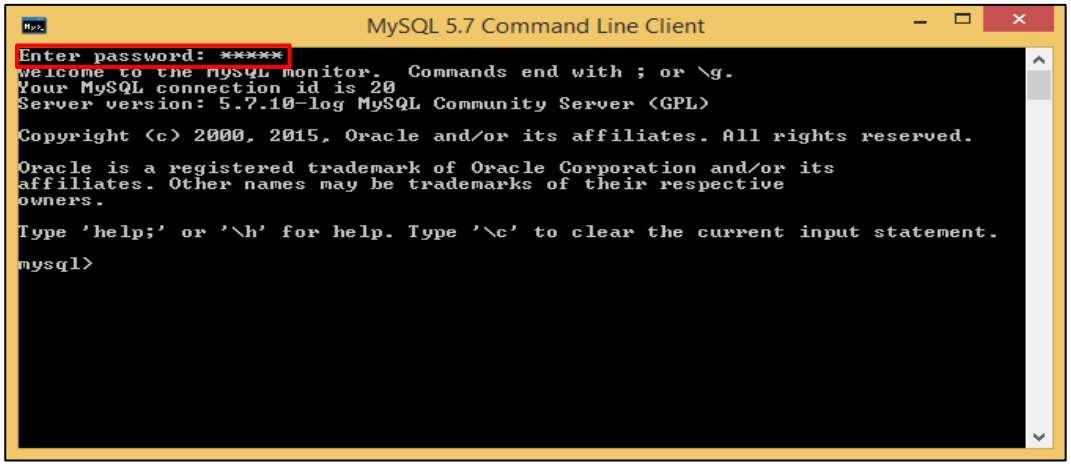


Test MySQL

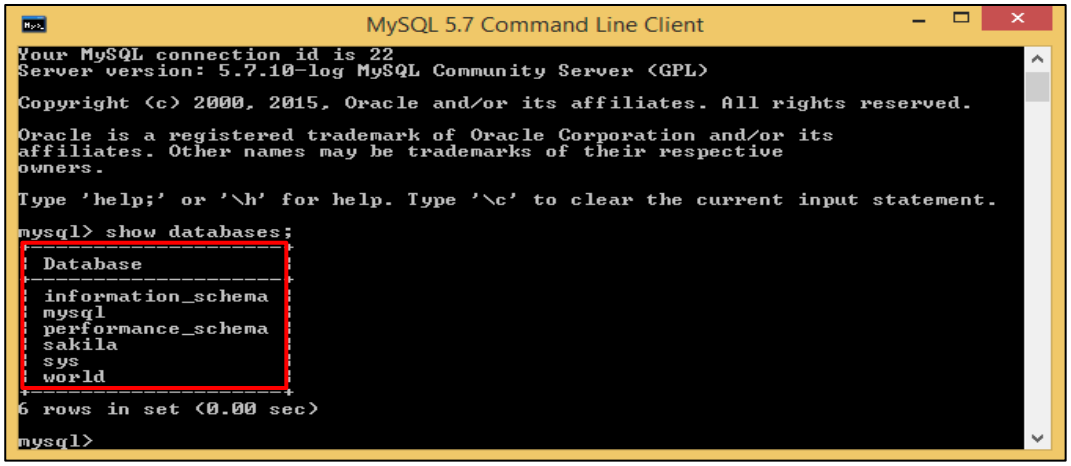
Start -> MySql Command Line Client

MySQL Root Password: admin





List all defined DBs: show databases;



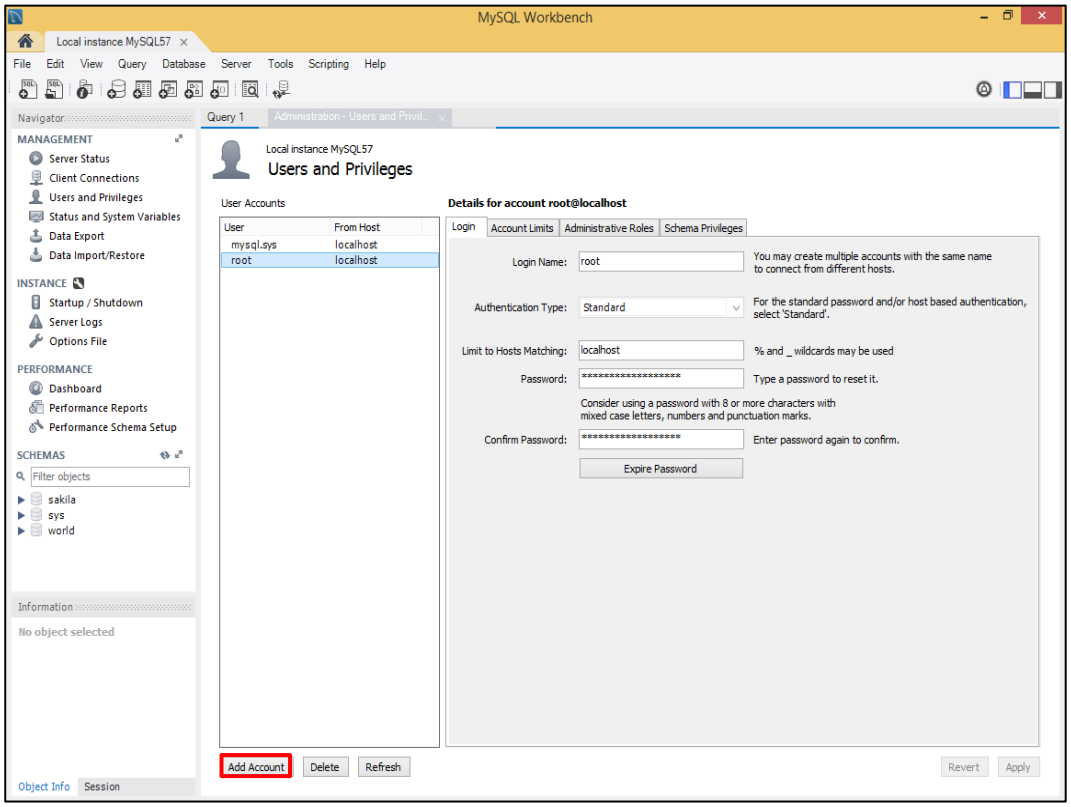
use sakila;

select * from actor;

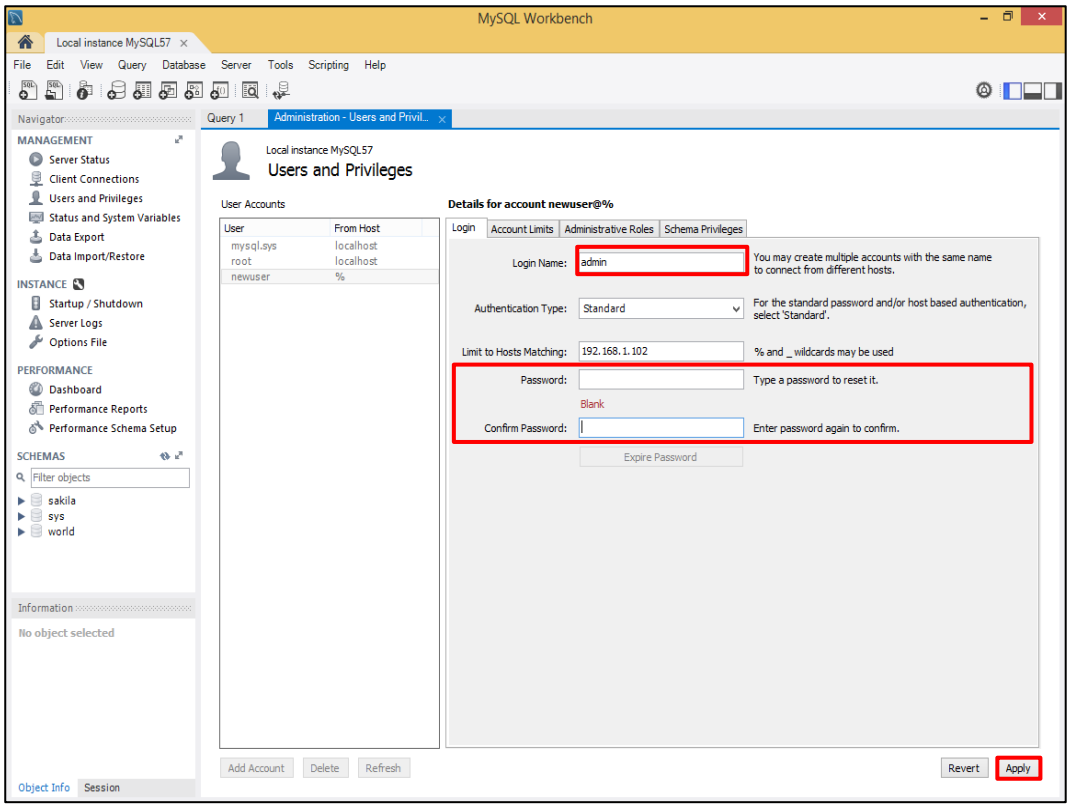
```
MySQL 5.7 Command Line Client
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)

mysql> use sakila;
Database changed
mysql> select * from actor;
+----------+------------+--------------+---------------------+
| actor_id | first_name | last_name    | last_update         |
+----------+------------+--------------+---------------------+
|        1 | PENELOPE   | GUINESS      | 2006-02-15 04:34:33 |
|        2 | NICK       | WAHLBERG     | 2006-02-15 04:34:33 |
|        3 | ED         | CHASE        | 2006-02-15 04:34:33 |
|        4 | JENNIFER   | DAVIS        | 2006-02-15 04:34:33 |
|        5 | JOHNNY     | LOLLOBRIGIDA | 2006-02-15 04:34:33 |
|        6 | BETTE      | NICHOLSON    | 2006-02-15 04:34:33 |
|        7 | GRACE      | MOSTEL       | 2006-02-15 04:34:33 |
|        8 | MATTHEW    | JOHANSSON    | 2006-02-15 04:34:33 |
|        9 | JOE        | SWANK        | 2006-02-15 04:34:33 |
|       10 | CHRISTIAN  | GABLE        | 2006-02-15 04:34:33 |
|       11 | ZERO       | CAGE         | 2006-02-15 04:34:33 |
|       12 | KARL       | BERRY        | 2006-02-15 04:34:33 |
|       13 | UMA        | WOOD         | 2006-02-15 04:34:33 |
|       14 | VIVIEN     | BERGEN       | 2006-02-15 04:34:33 |
|       15 | CUBA       | OLIVIER      | 2006-02-15 04:34:33 |
```

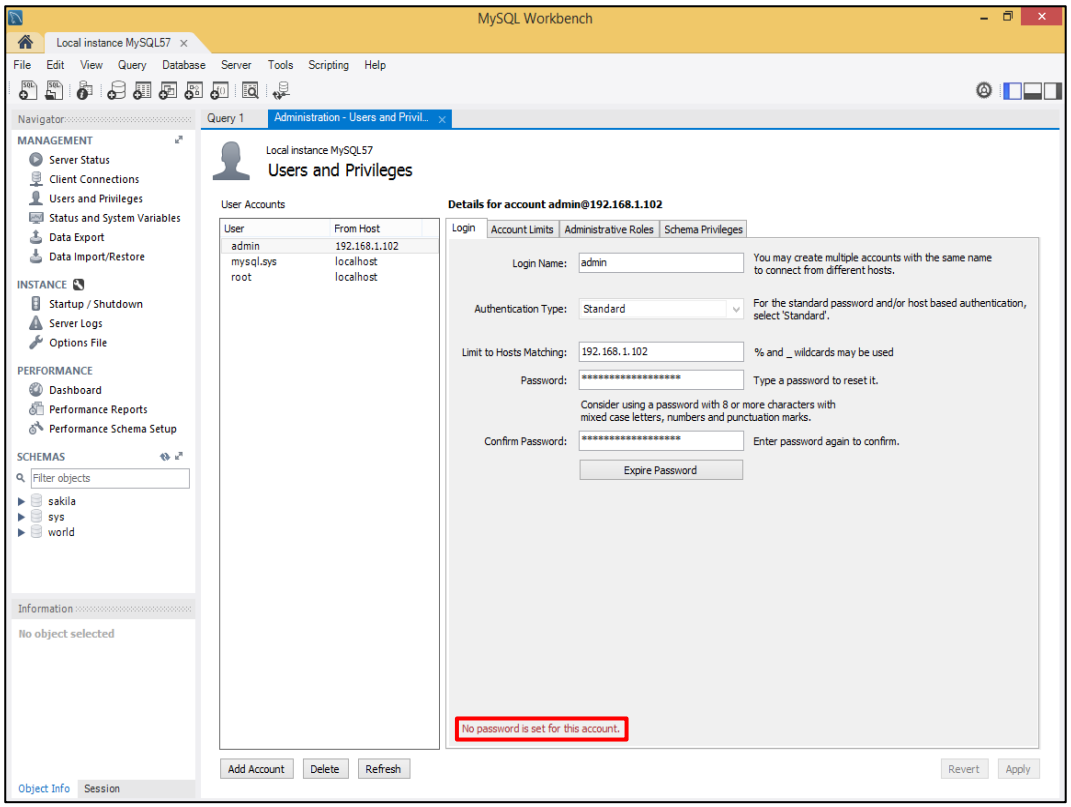
Give a Remote Computer access to MySQL

To give authorization to another computer to access any DB, please start MySql Workbench. Go to "Users and Privileges". Click on the "Add Account" Button and fill in the requested data, which is needed to get an access. Leave the password blank.

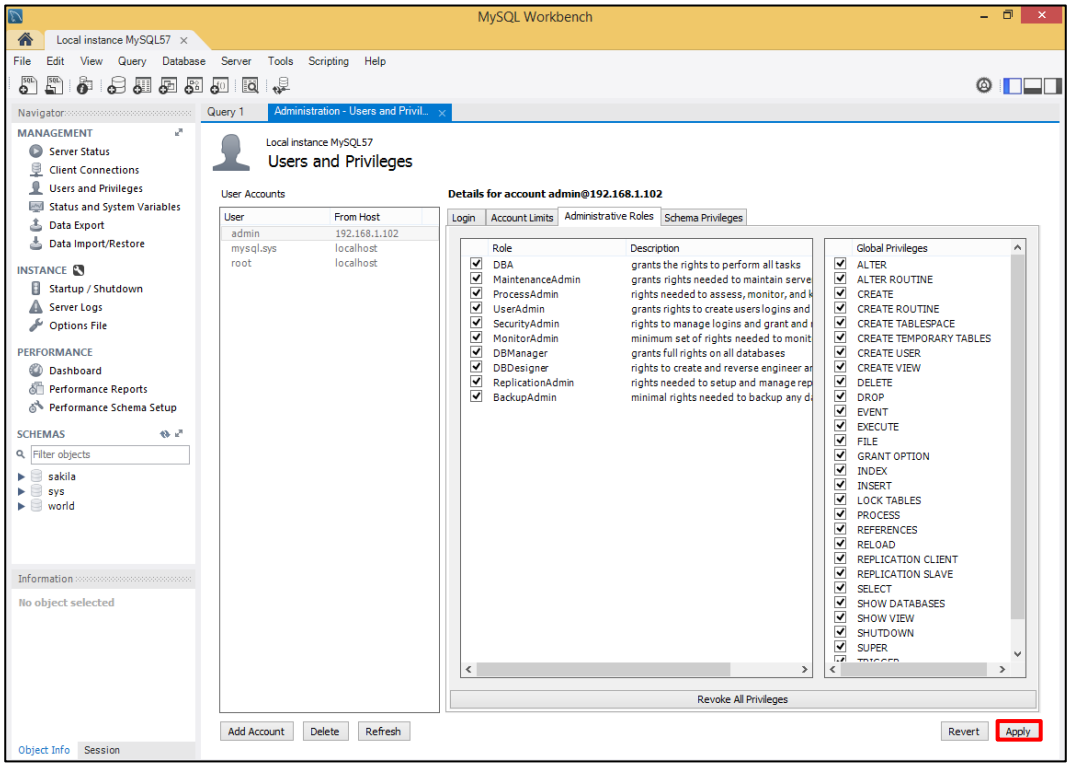


Login Name: admin
Password: admin

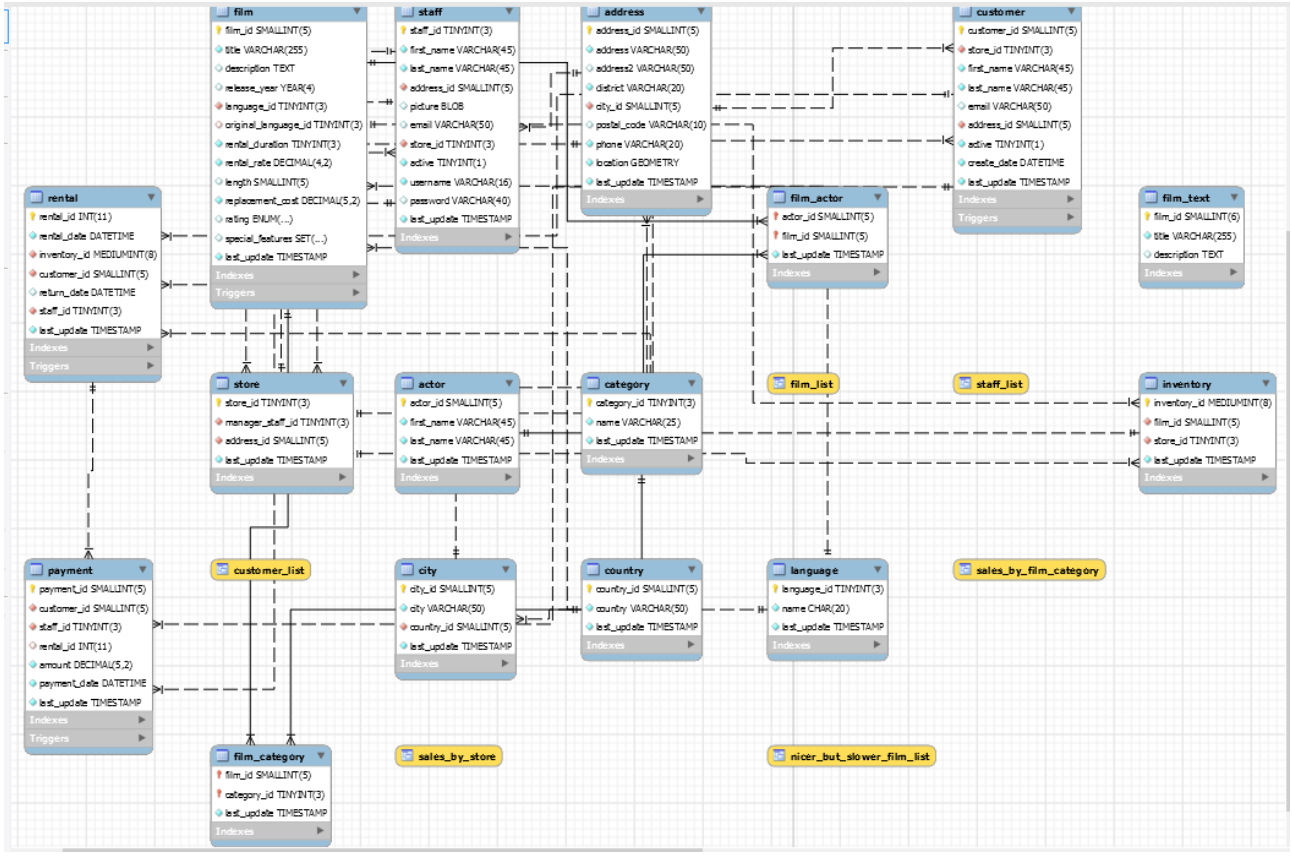

MySQL Workbench
Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
SCHEMAS
Filter objects
sakila
sys
world
Information
No object selected
Object Info
Session
Query 1
Administration - Users and Privil...
Local instance MySQL57
Users and Privileges
User Accounts
User
From Host
mysql.sys
localhost
root
localhost
newuser
%
Details for account newuser@%
Login
Account Limits
Administrative Roles
Schema Privileges
Login Name:
admin
You may create multiple accounts with the same name to connect from different hosts.
Authentication Type:
Standard
For the standard password and/or host based authentication, select 'Standard'.
Limit to Hosts Matching:
192.168.1.102
% and _ wildcards may be used
Password:
Type a password to reset it.
Blank
Confirm Password:
Enter password again to confirm.
Expire Password
Add Account
Delete
Refresh
Revert
Apply



After you have created an account go to "Administrative Roles" and give the account the needed rights.



This is the ERD of Sakila, which is one of the default databases from the MySQL Workbench.



Download MySQL JDBC driver

https://dev.mysql.com/downloads/connector/j/

Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!

Generally Available (GA) Releases | Development Releases

Connector/J 5.1.45

Select Operating System:
Platform Independent

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-java-5.1.45.tar.gz)	5.1.45	3.3M	Download MD5: ab9ac454a959859a297b53bdbf156f3c | Signature
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-java-5.1.45.zip)	5.1.45	3.6M	Download MD5: 2e3a933f8f7642345d5f34bf43374f8e | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

Begin Your Download

mysql-connector-java-5.1.45.zip

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

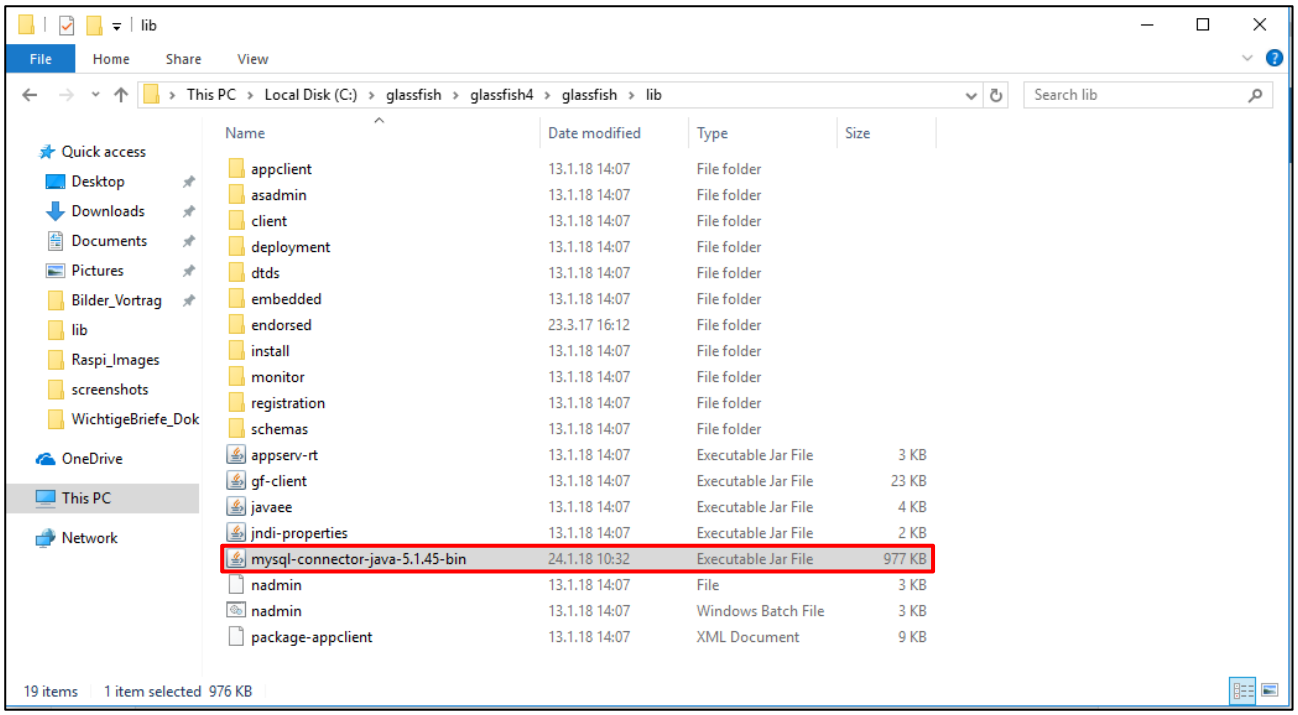
- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system
- Comment in the MySQL Documentation

Login » using my Oracle Web account | Sign Up » for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

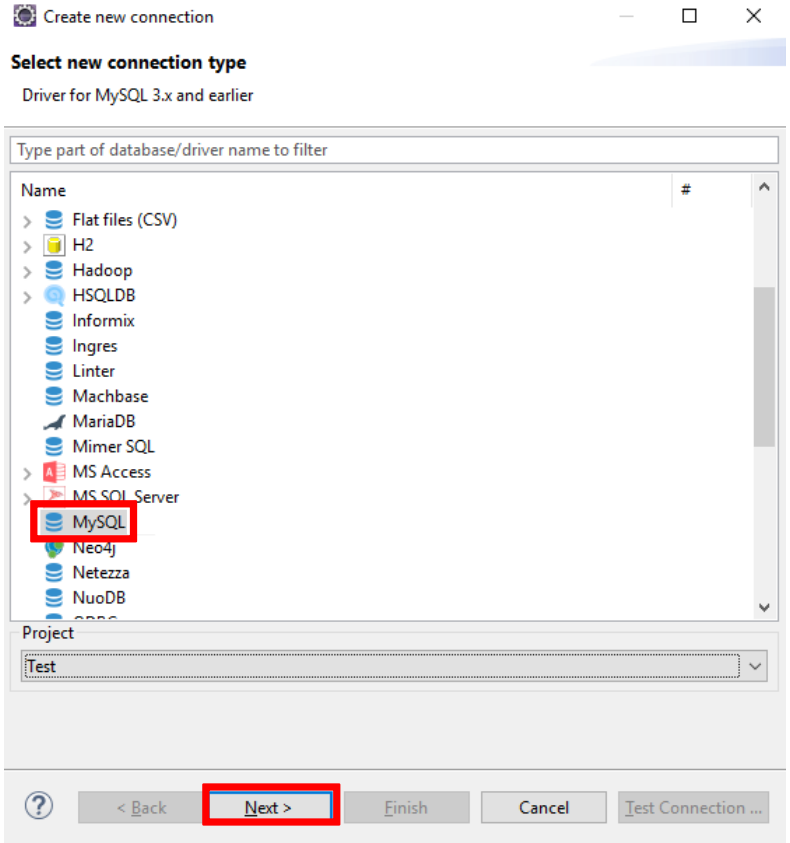
Download the zip file and extract it. Then move the connector (mysql-connector-java-5.1.xxx-bin) to C:\glassfish\glassfish4\glassfish\lib Otherwise it won't work.

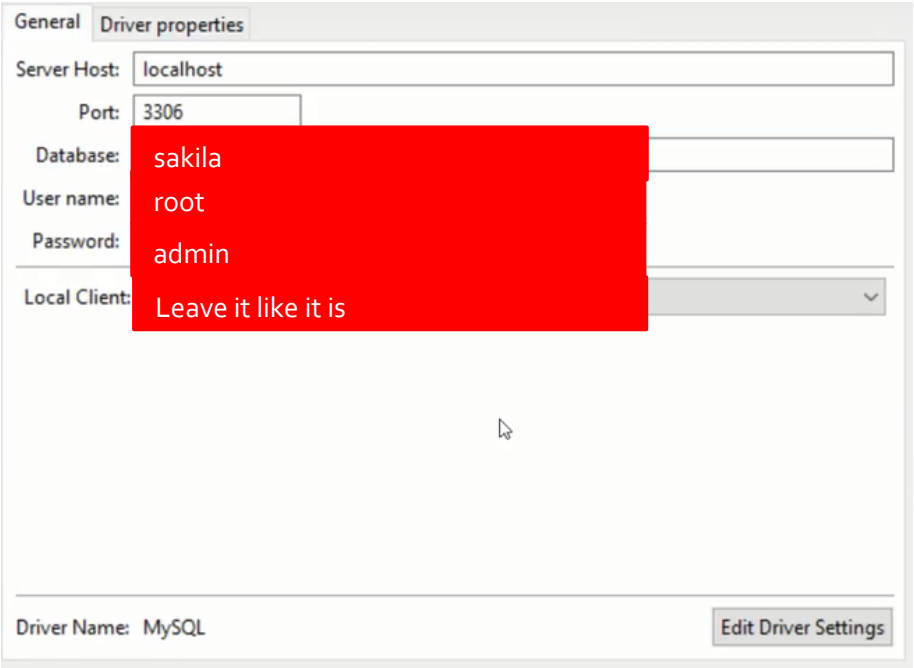

lib
File
Home
Share
View
This PC › Local Disk (C:) › glassfish › glassfish4 › glassfish › lib
Search lib
Quick access
Desktop
Downloads
Documents
Pictures
Bilder_Vortrag
lib
Raspi_Images
screenshots
WichtigeBriefe_Dok
OneDrive
This PC
Network
Name
Date modified
Type
Size
appclient 13.1.18 14:07 File folder
asadmin 13.1.18 14:07 File folder
client 13.1.18 14:07 File folder
deployment 13.1.18 14:07 File folder
dtds 13.1.18 14:07 File folder
embedded 13.1.18 14:07 File folder
endorsed 23.3.17 16:12 File folder
install 13.1.18 14:07 File folder
monitor 13.1.18 14:07 File folder
registration 13.1.18 14:07 File folder
schemas 13.1.18 14:07 File folder
appserv-rt 13.1.18 14:07 Executable Jar File 3 KB
gf-client 13.1.18 14:07 Executable Jar File 23 KB
javaee 13.1.18 14:07 Executable Jar File 4 KB
jndi-properties 13.1.18 14:07 Executable Jar File 2 KB
mysql-connector-java-5.1.45-bin 24.1.18 10:32 Executable Jar File 977 KB
nadmin 13.1.18 14:07 File 3 KB
nadmin 13.1.18 14:07 Windows Batch File 3 KB
package-appclient 13.1.18 14:07 XML Document 9 KB
19 items
1 item selected 976 KB

Create a new MySQL Connection

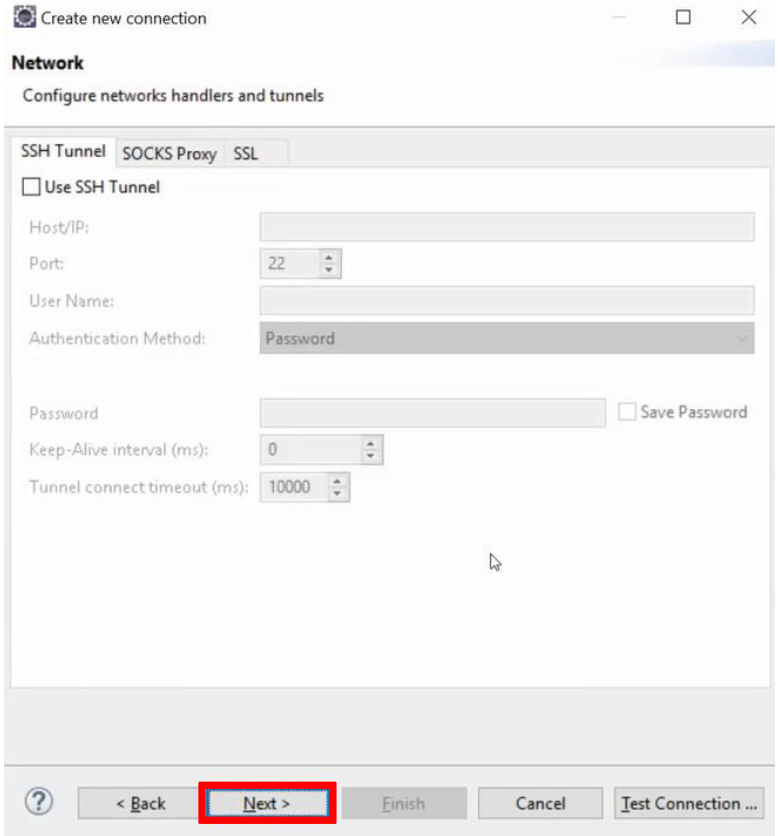
Go to Database -> New Connection

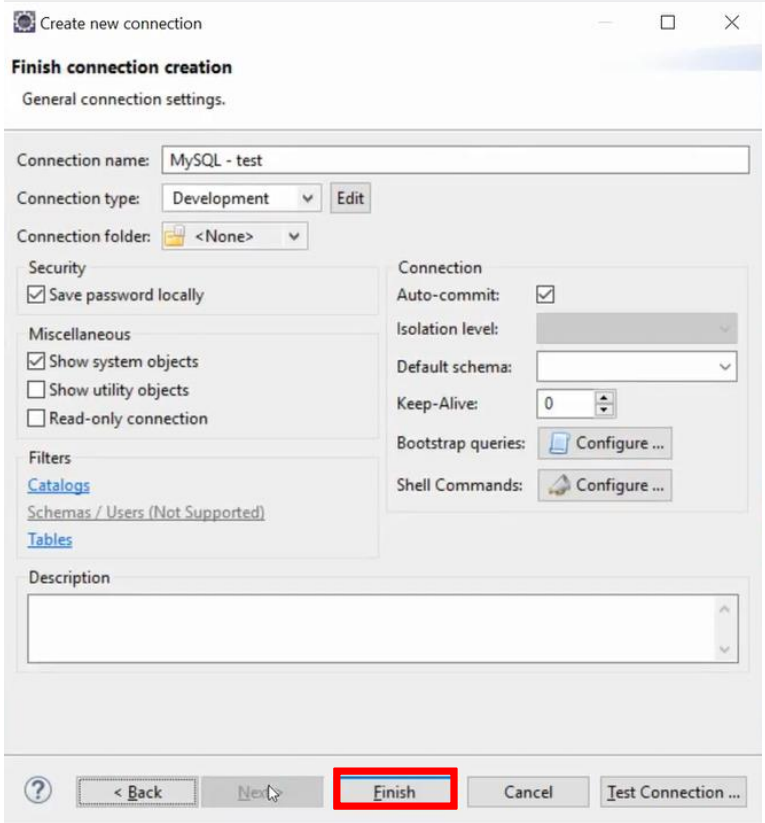
Choose MySQL from the List and press Next





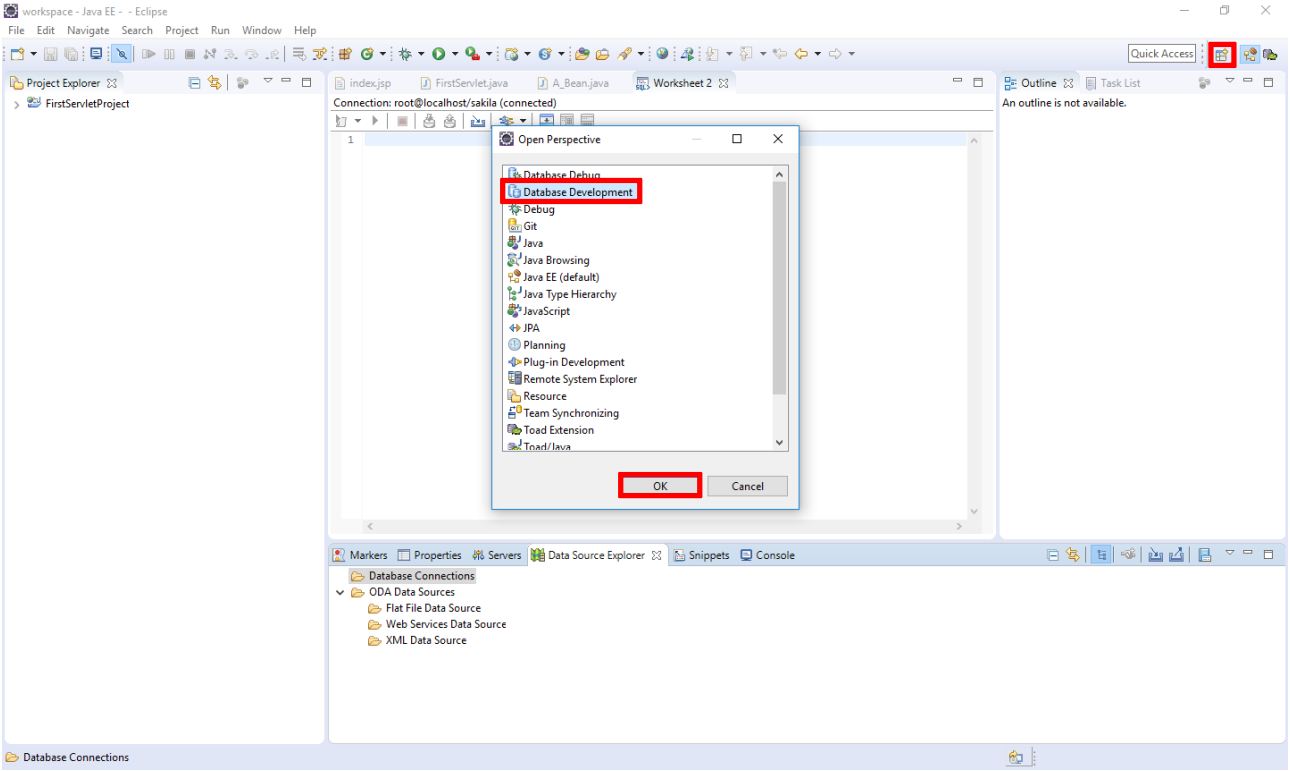
Afterwards press Test Connection and Next


Create new connection
Network
Configure networks handlers and tunnels
SSH Tunnel
SOCKS Proxy
SSL
Use SSH Tunnel
Host/IP:
Port:
22
User Name:
Authentication Method:
Password
Password
Save Password
Keep-Alive interval (ms):
0
Tunnel connect timeout (ms):
10000
< Back
Next >
Finish
Cancel
Test Connection ...

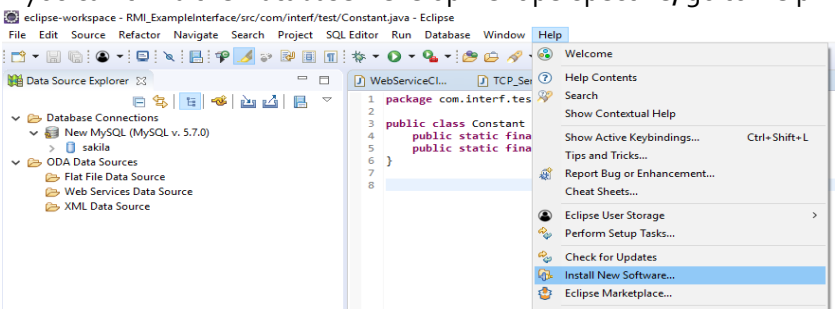

Create new connection
Finish connection creation
General connection settings.
Connection name:
MySQL - test
Connection type:
Development
Edit
Connection folder:
<None>
Security
Save password locally
Miscellaneous
Show system objects
Show utility objects
Read-only connection
Filters
Catalogs
Schemas / Users (Not Supported)
Tables
Connection
Auto-commit:
Isolation level:
Default schema:
Keep-Alive:
0
Bootstrap queries:
Configure ...
Shell Commands:
Configure ...
Description
< Back
Finish
Cancel
Test Connection ...

Create DB-Connection with DBeaver

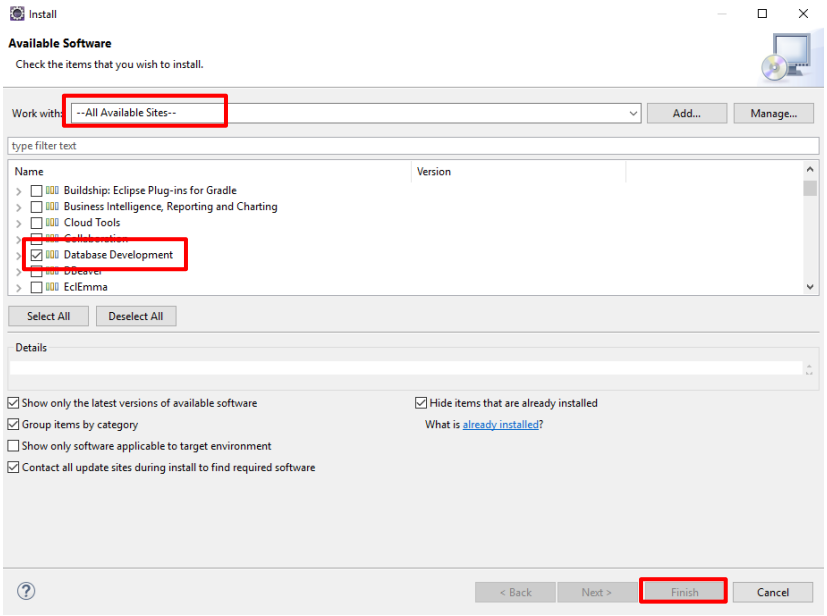
Add new Database Development Interface



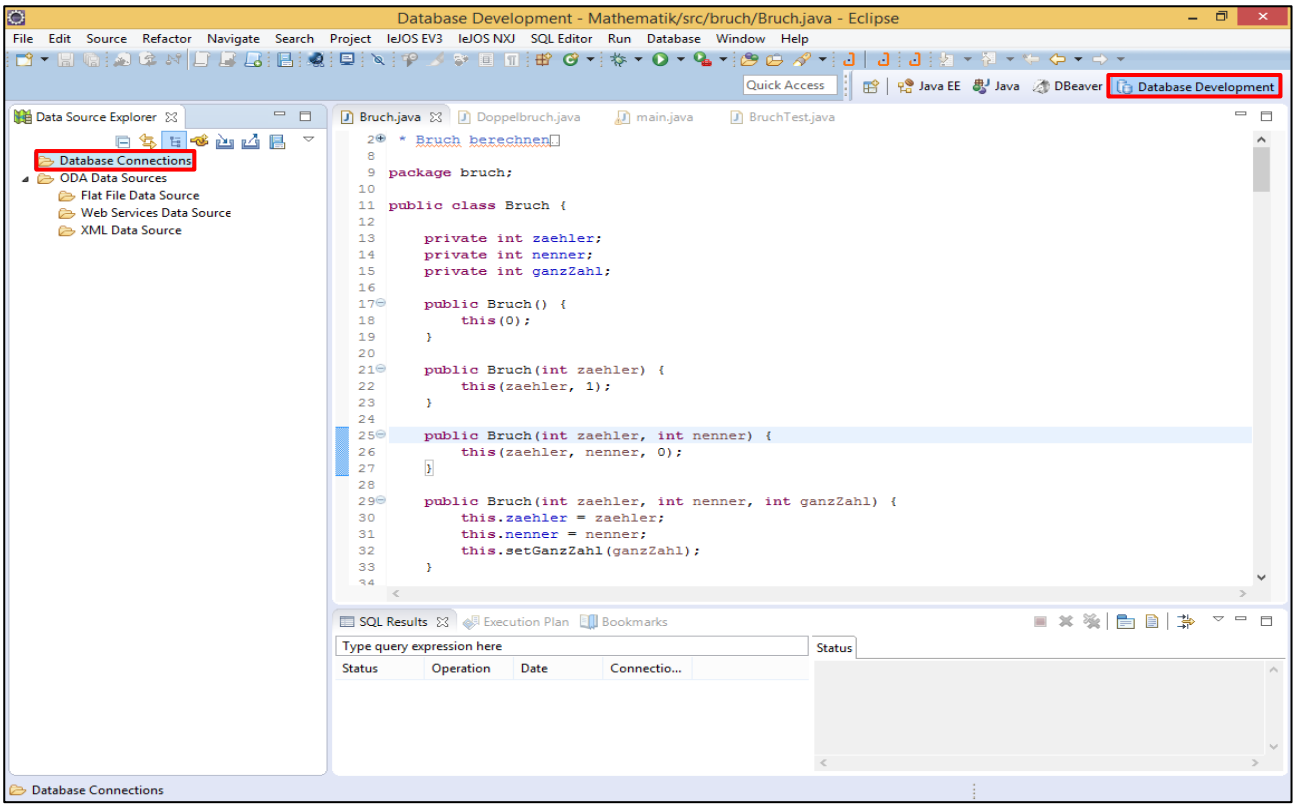
If you can't find the Database Development perspective, go to Help -> Install New Software

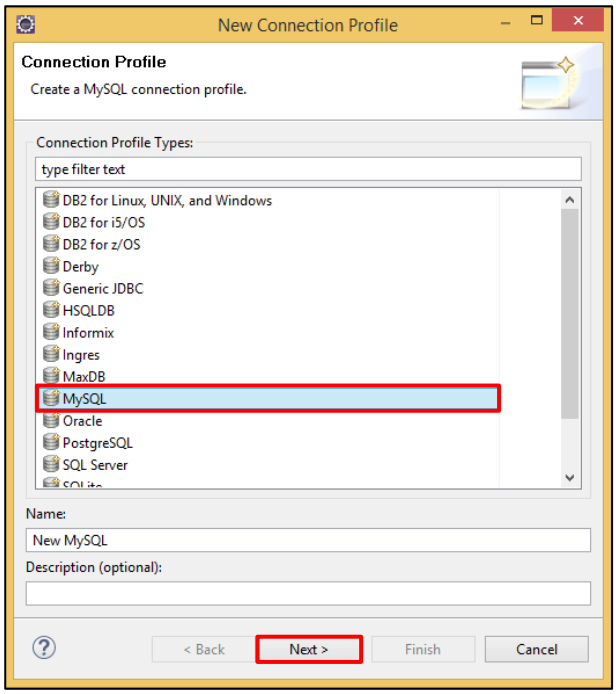


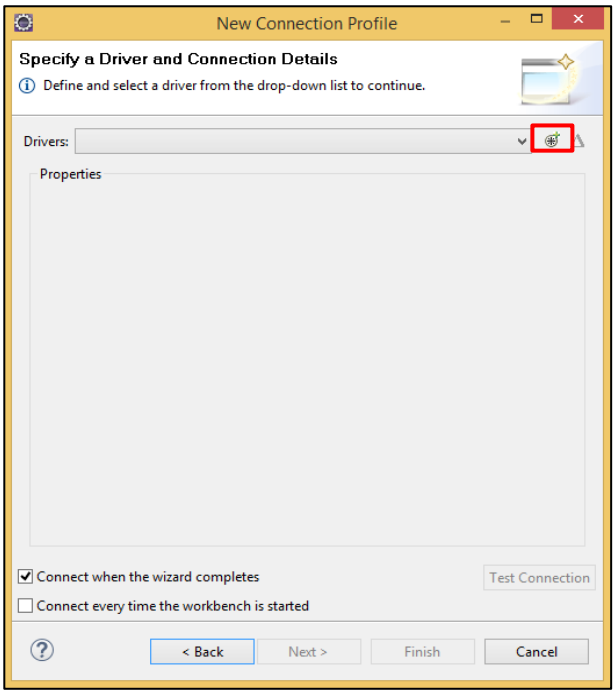
Then choose: "—All Available Sites--" and start the search



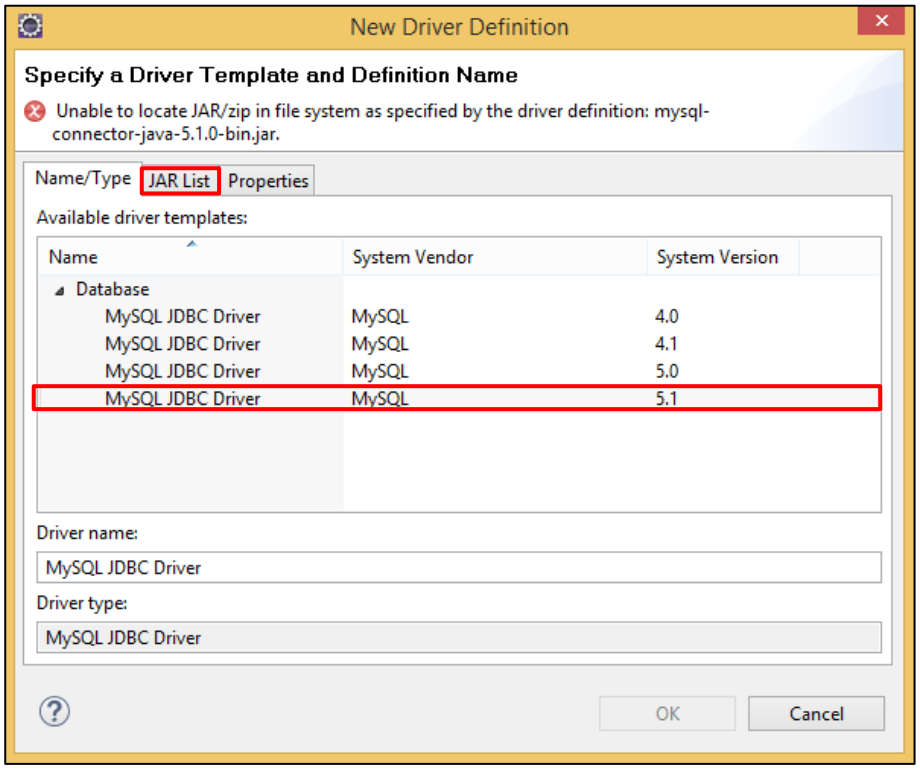
Select Database Development and press Finish
Right Mouse-Click on Database Connections → new....

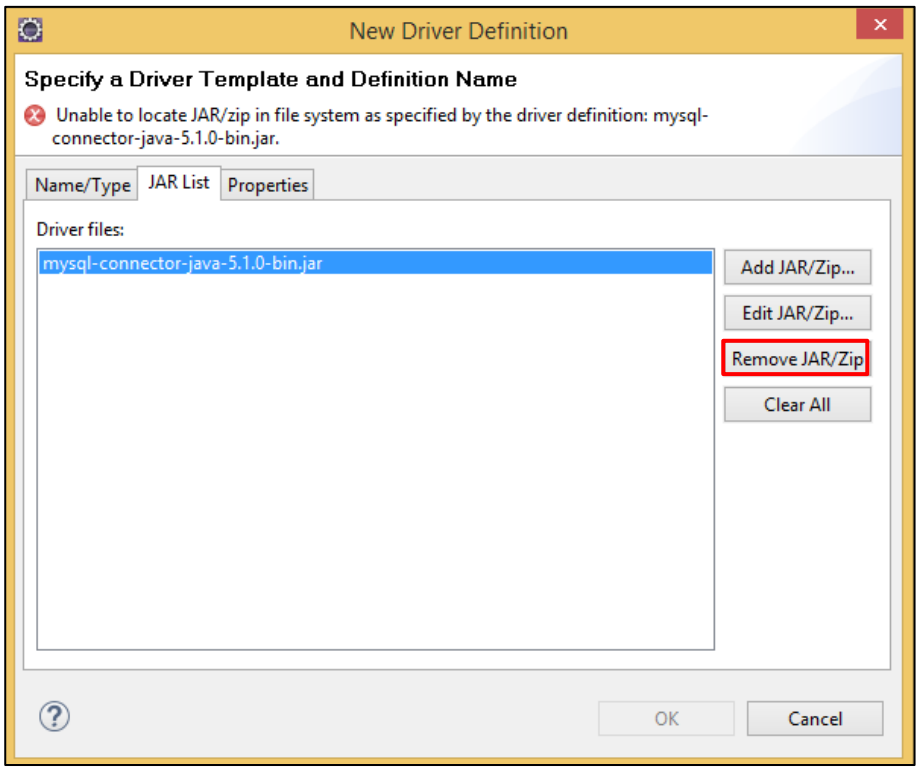



New Connection Profile
Connection Profile
Create a MySQL connection profile.
Connection Profile Types:
type filter text
DB2 for Linux, UNIX, and Windows
DB2 for i5/OS
DB2 for z/OS
Derby
Generic JDBC
HSQLDB
Informix
Ingres
MaxDB
MySQL
Oracle
PostgreSQL
SQL Server
Name:
New MySQL
Description (optional):
< Back
Next >
Finish
Cancel

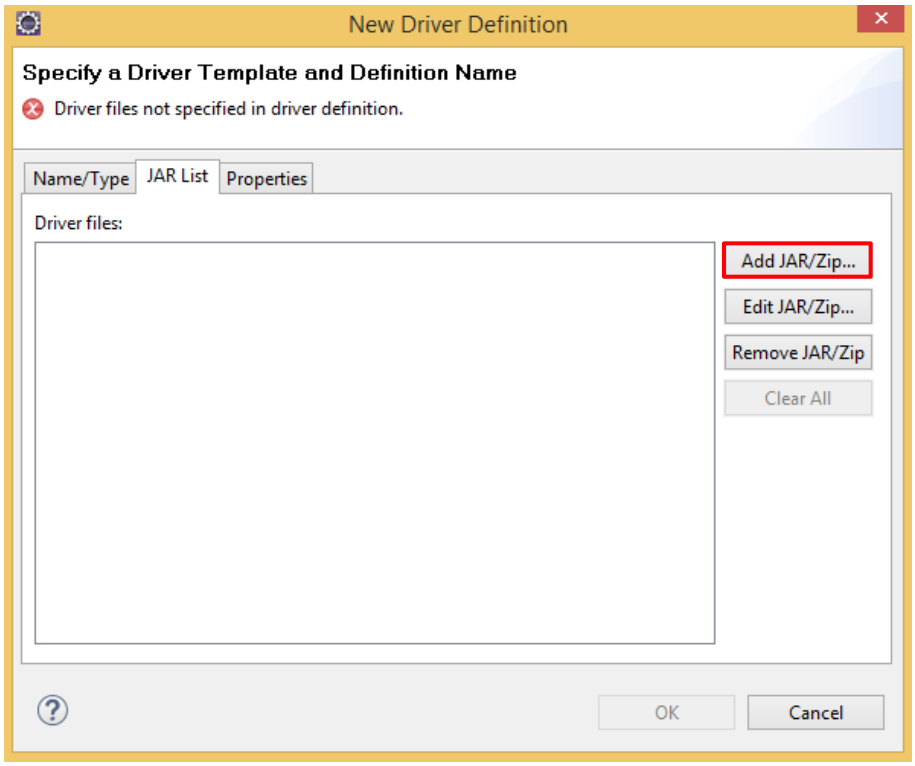

New Connection Profile
Specify a Driver and Connection Details
Define and select a driver from the drop-down list to continue.
Drivers:
Properties
Connect when the wizard completes
Connect every time the workbench is started
Test Connection
< Back
Next >
Finish
Cancel

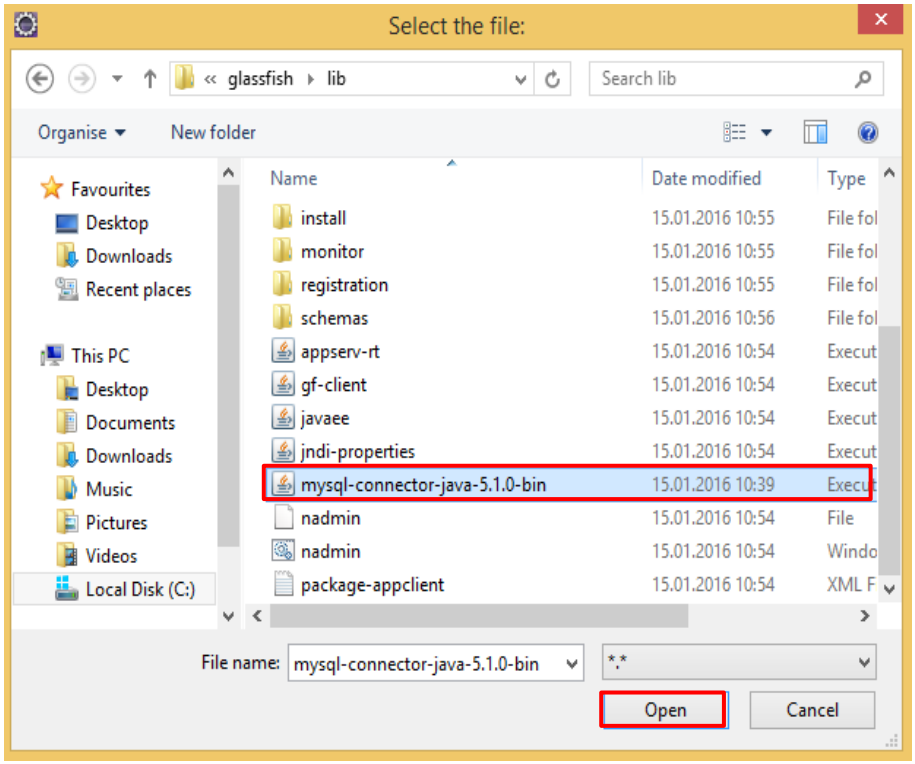
Select the MySQL JDBC Driver with the matching System Version (January 2015: 5.1) and select the «Jar List»-Tab. Remove the current Jar in the list and click on "Add JAR/Zip...".

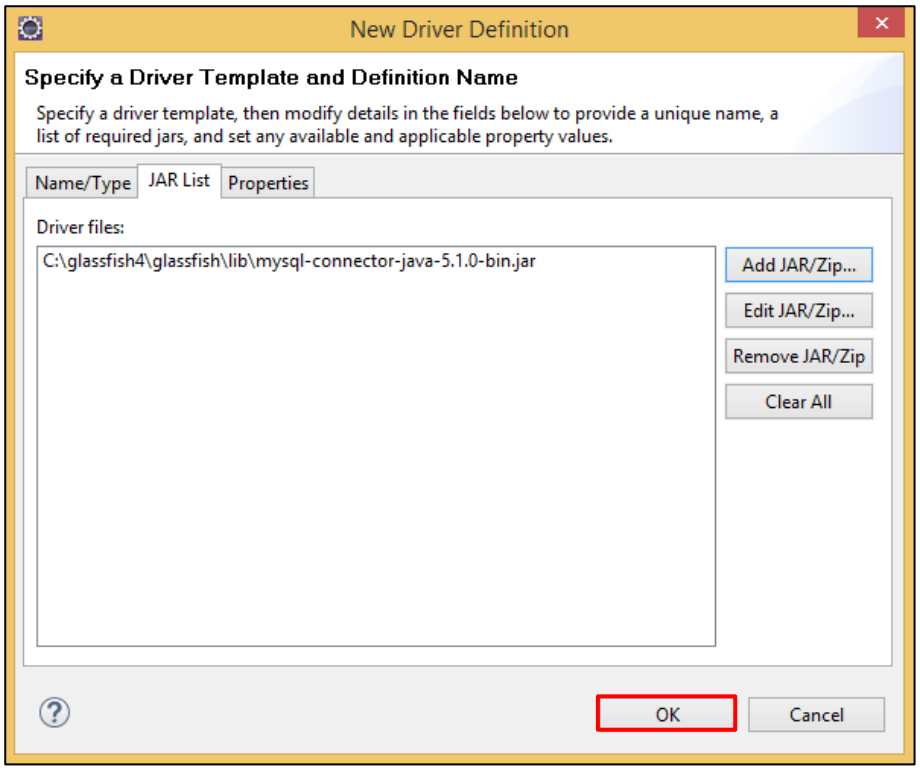




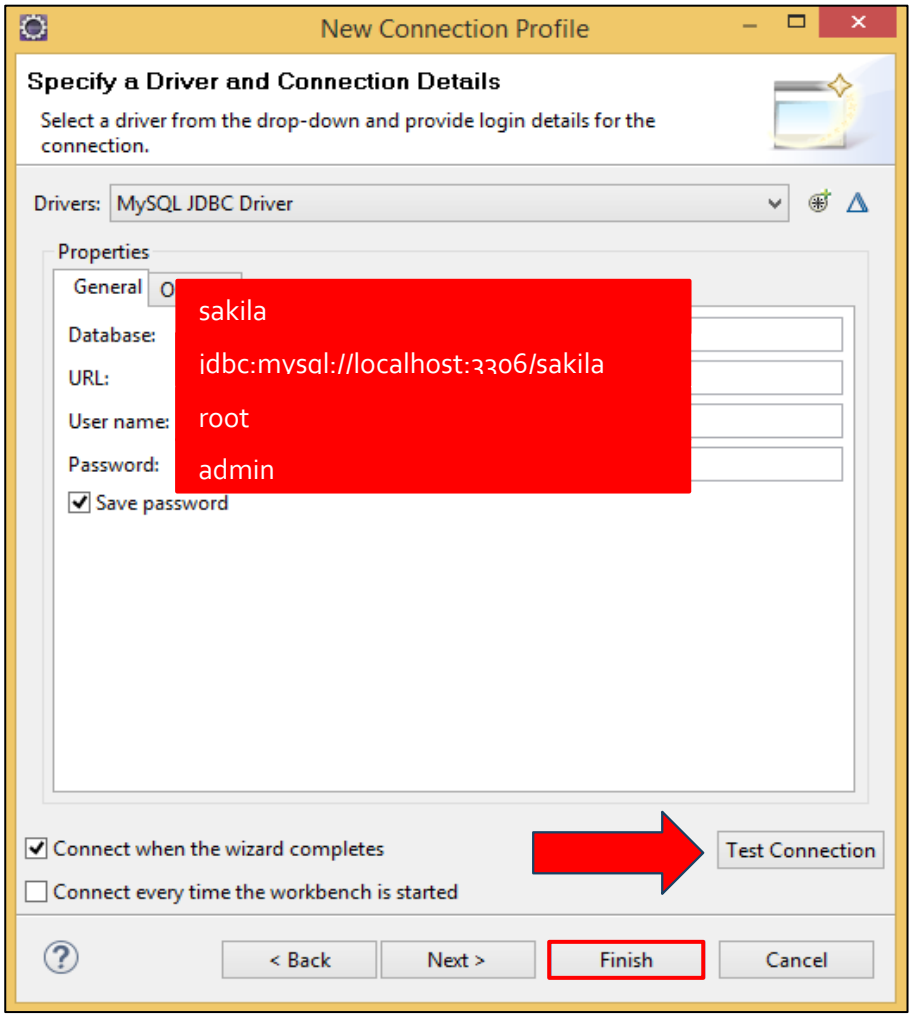
Select "Add JAR/Zip". Then navigate to the installation directory of the glassfish-server and select the driver in the directory .../glassfish/lib/ select mysql-connector-java-5.1.0-bin and click on "Open".







MySQL Root Password: admin



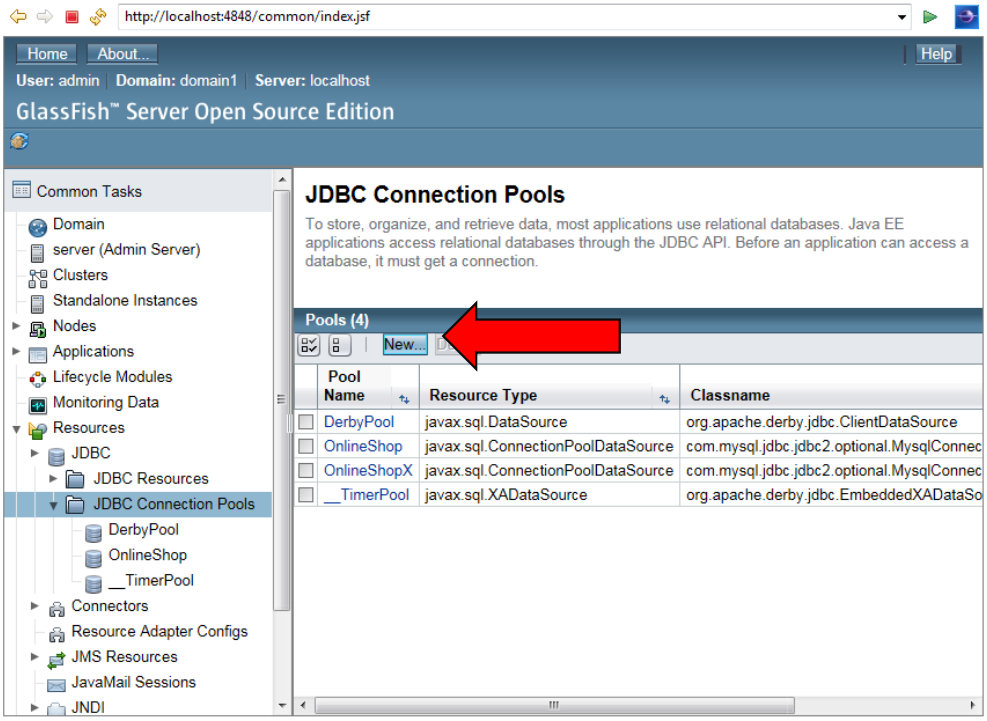
Basic MySql commands

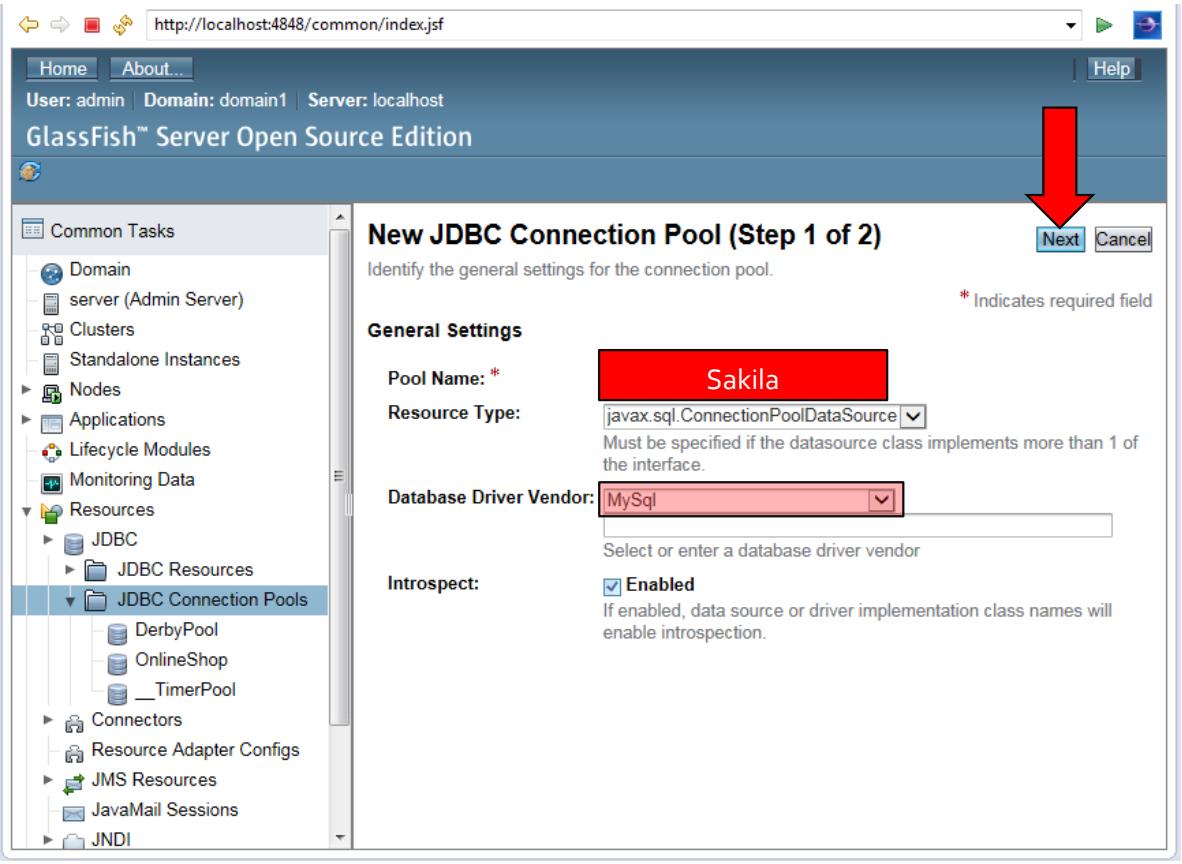
Start -> MySql Command Line Client

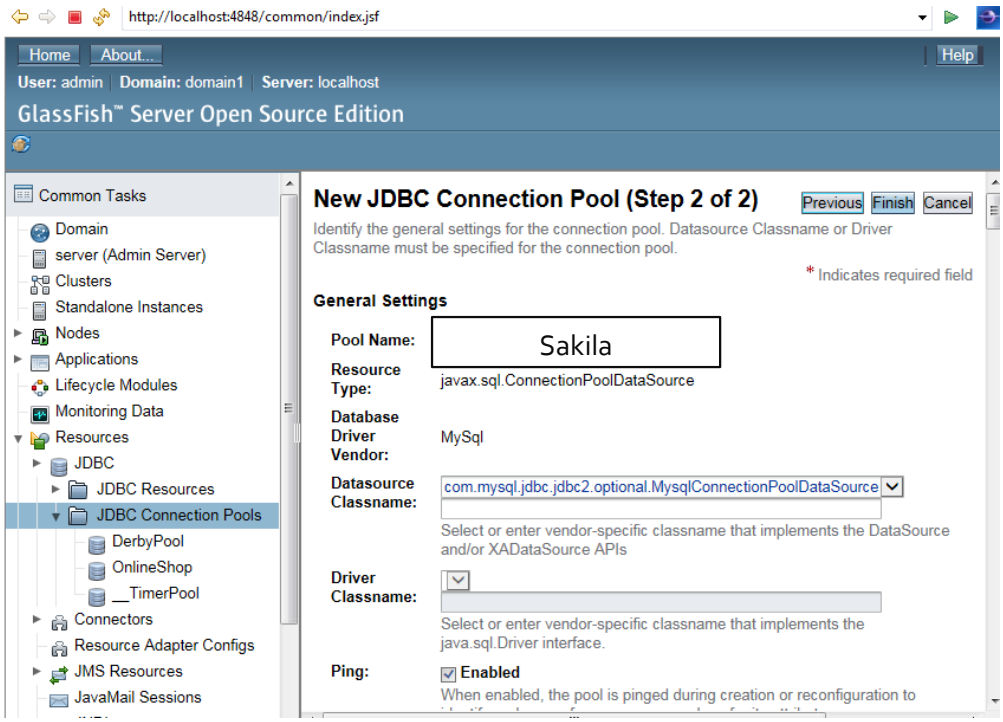
Command	Description
Show databases;	Shows you the already defined DBs.
Use test;	To use a DB.
Show tables;	Shows you all the tables from the using DB.
Create table Classmates (*Id int PRIMARY KEY NOT NULL,* *fname varchar (25),* *lname varchar (25)* *);* *CREATE UNIQUE INDEX PRIMARY ON CLASSMATES (ID);* *INSERT INTO classmates (id, lname, fname)* *VALUES (1, 'David', 'Etter');*	It creates a table with the name *Classmates". The table contains a primary key and to columns for the first- and lastname.

Setup a DB Connection via JNDI on the Application Server

Start the Glassfish Admin-Console and go to **Resources** -> **JDBC** -> **JDBC ConnectionPool** -> **New…**

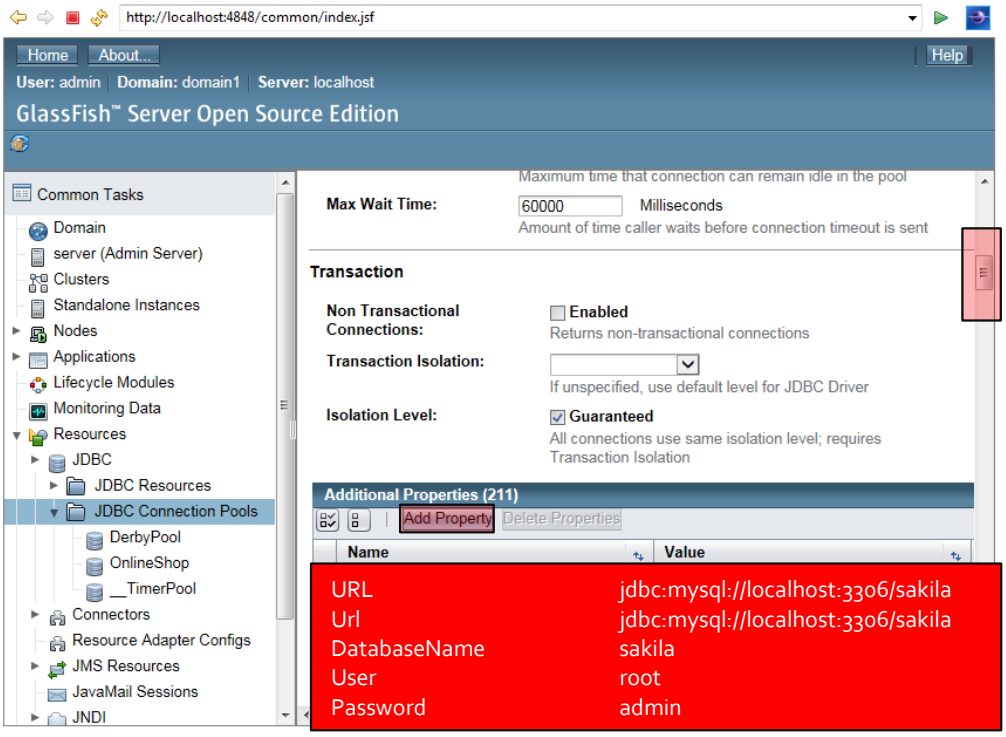






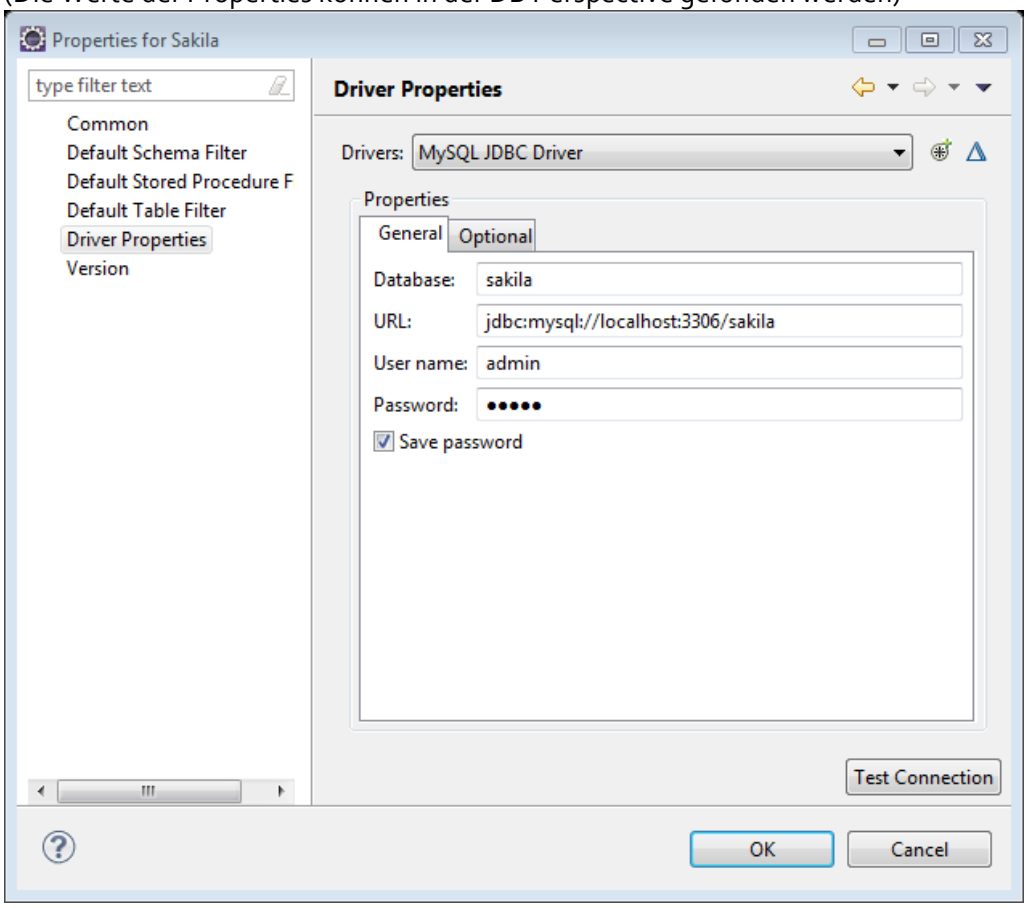
Datasource Classname: com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource

Scroll to the end of the page and set these propertes:
User → root
Password → admin
DatabaseName → sakila
Url → jdbc:mysql://localhost:3306/sakila
URL → jdbc:mysql://localhost:3306/sakila

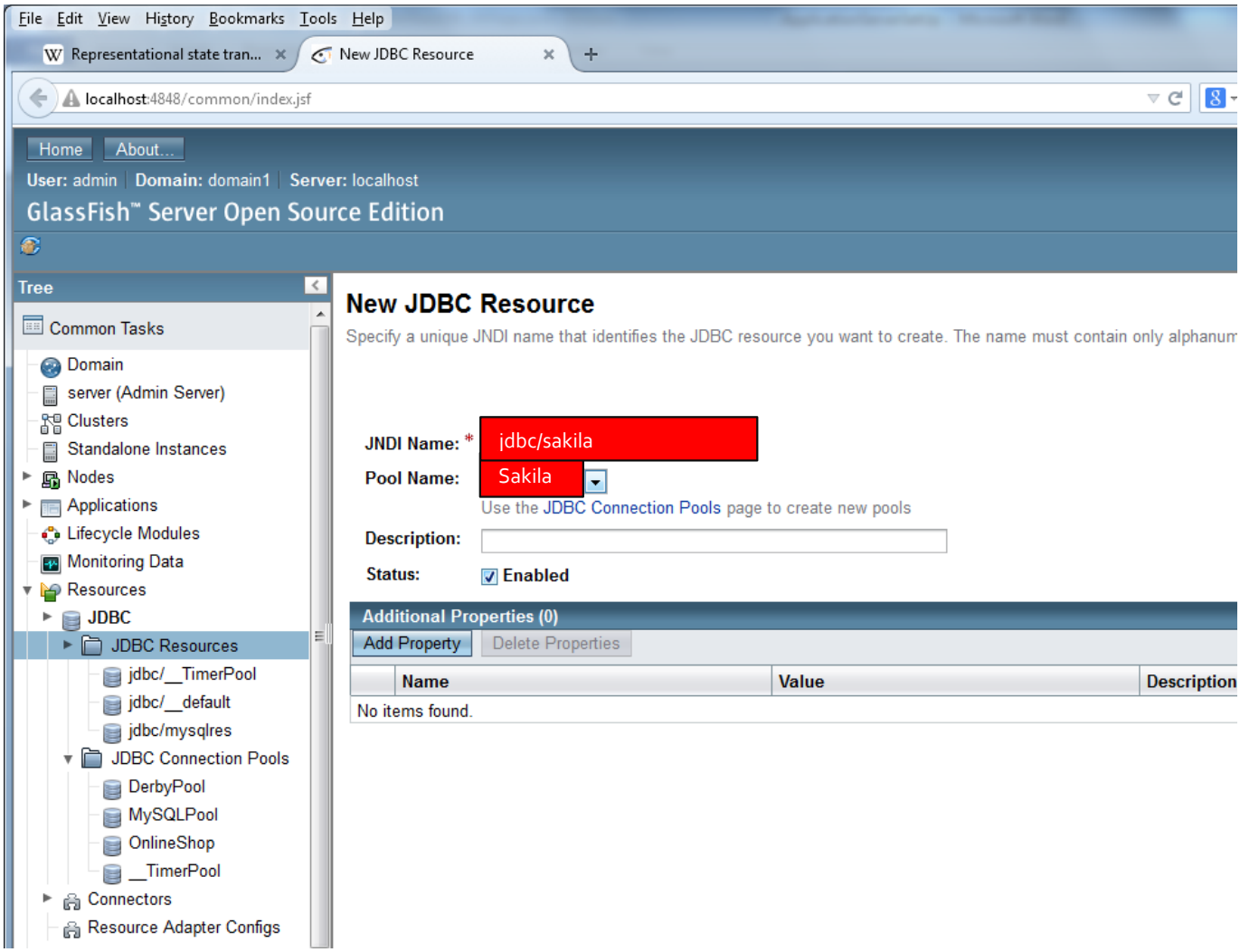


Simply press Finish at the end of the page.

(Die Werte der Properties können in der DB Perspective gefunden werden)



Define a new JNDI, that references our DB Connection:
JDBC Resources → New….

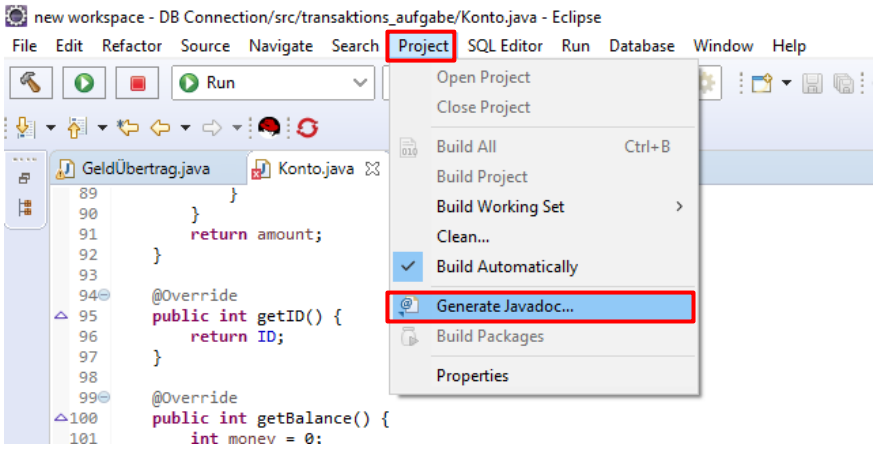


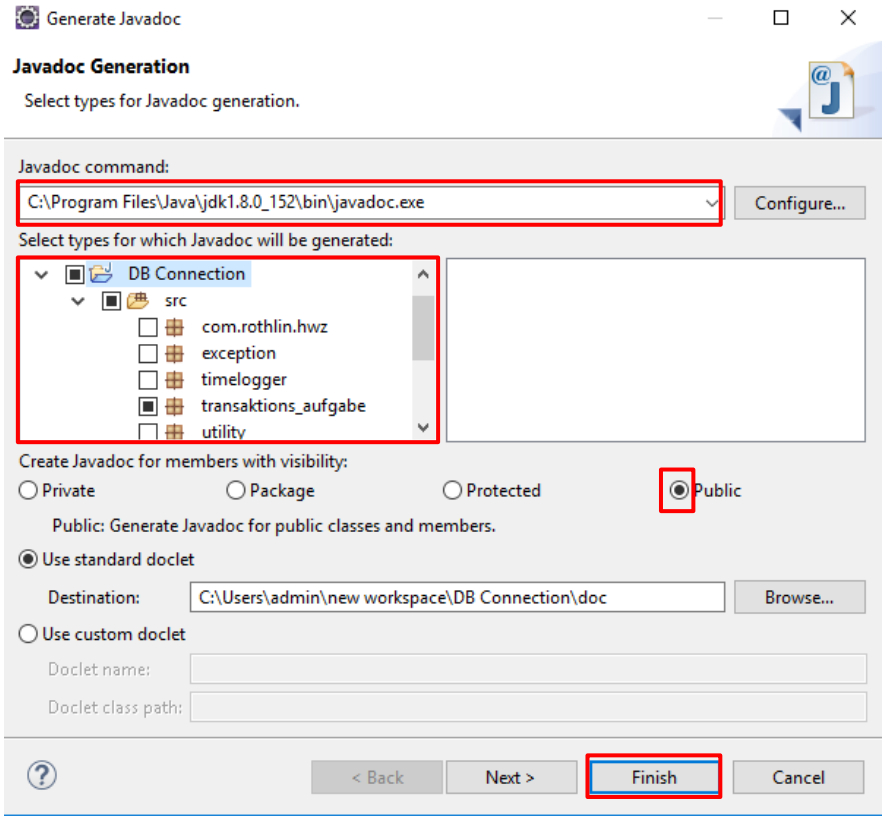
Simply press OK when you are done

Javadoc

Generating Javadoc

Click on *Project → Generate Javadoc…*





On the Javadoc command field, type: *C:\Program Files\Java\jdk1.8.0_152\bin\javadoc.exe*

Choose the project and the package where you'd like to generate the Javadoc.

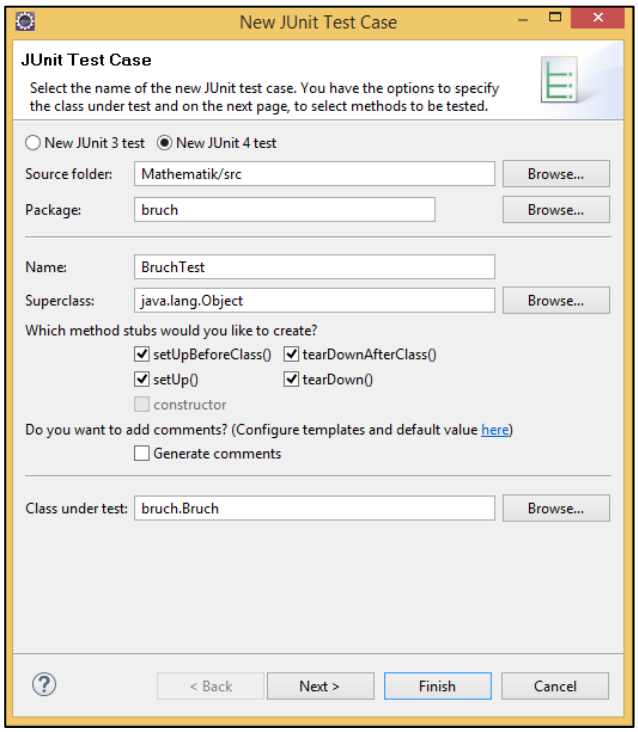
Be sure to set the visibility to **Public.**

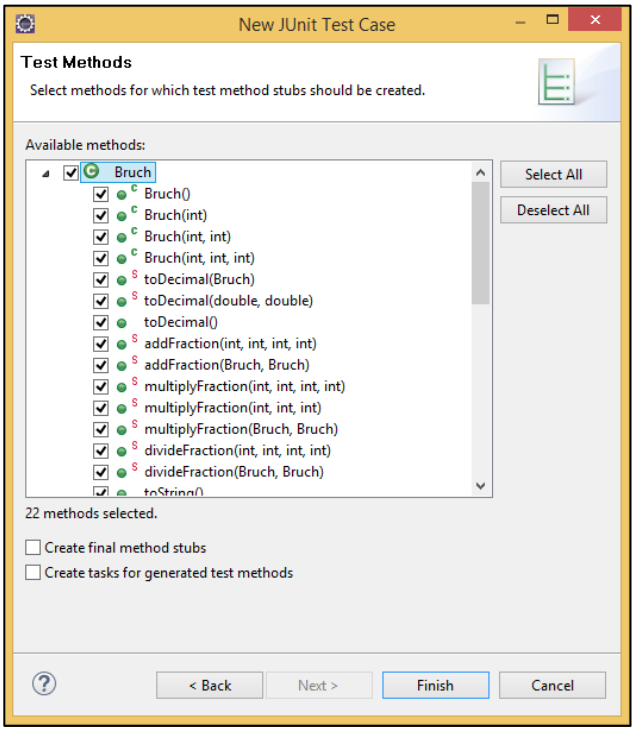
Click *Finish* when you are done.

Junit

Create Junit Test

open project -> right click on class -> New -> JUnit Test Case

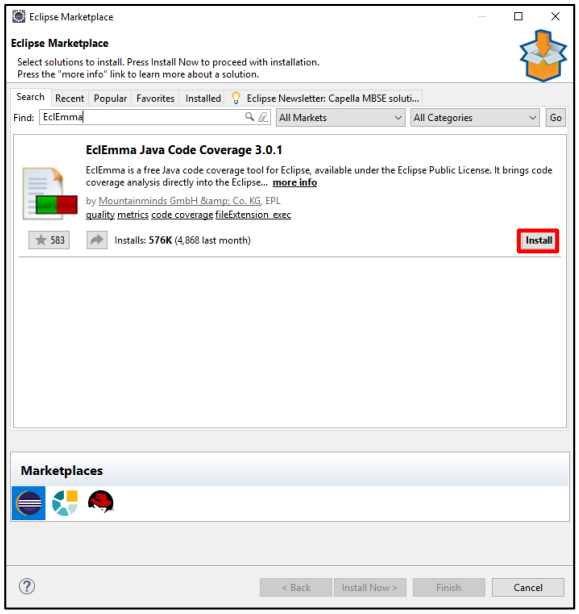


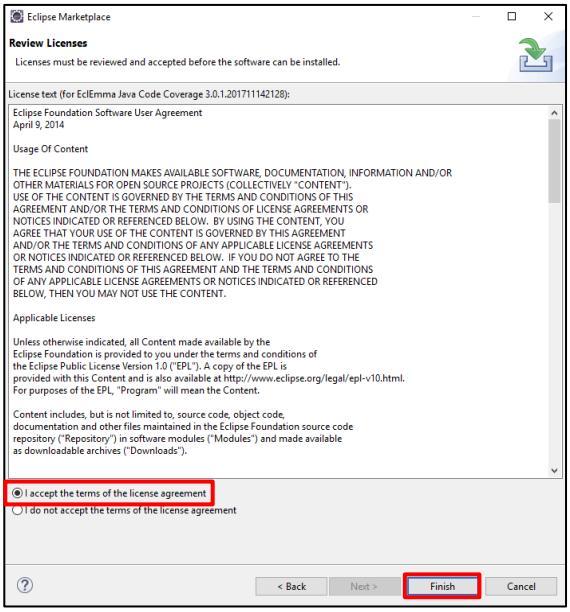


When JUnit 4 is not on the build path follow these steps:
click on project -> press Alt + Enter -> Java Build Path -> Libraries -> Add External JARs -> JUnit 4
Implementing Tests and run it as JUnit Test

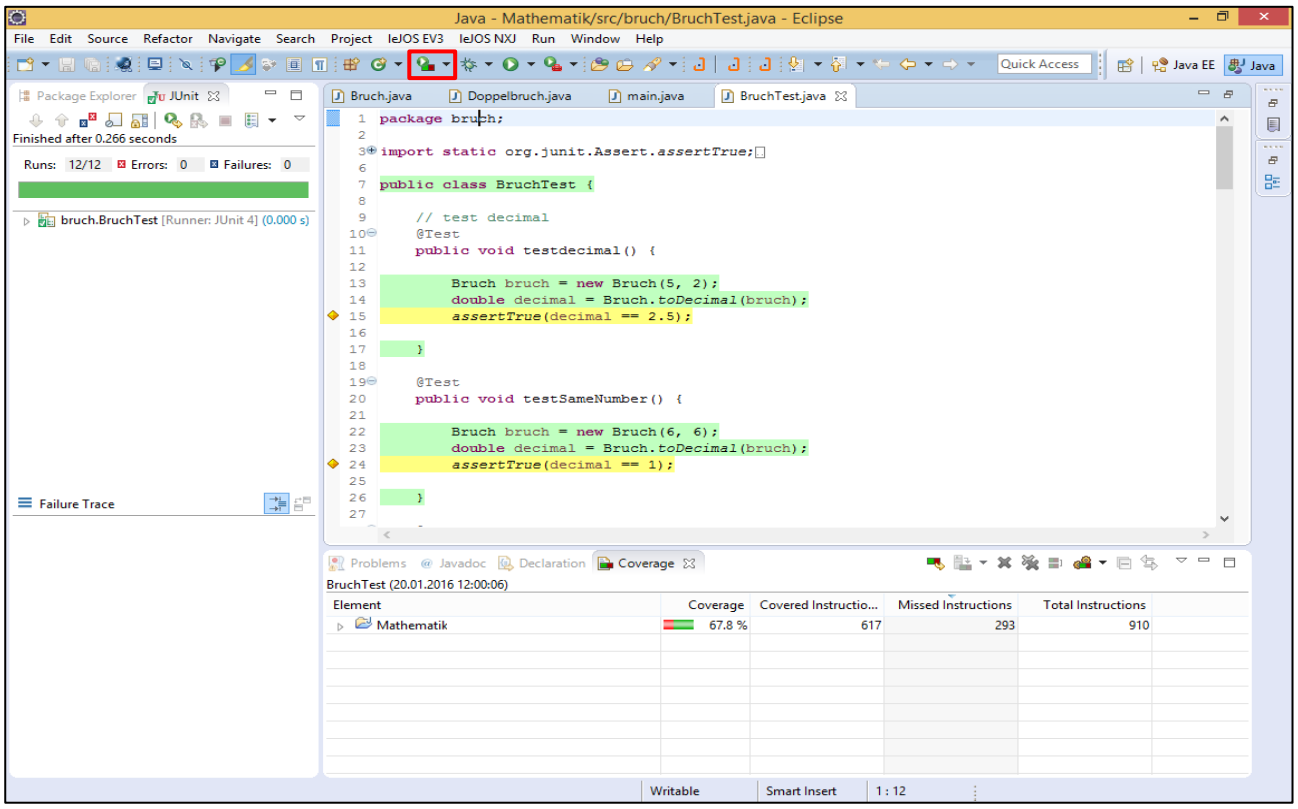
Coverage - Install EclEmma Plugin

```
Help -> Marketplace -> EclEmma
```





Coverage - Run Test with coverage report

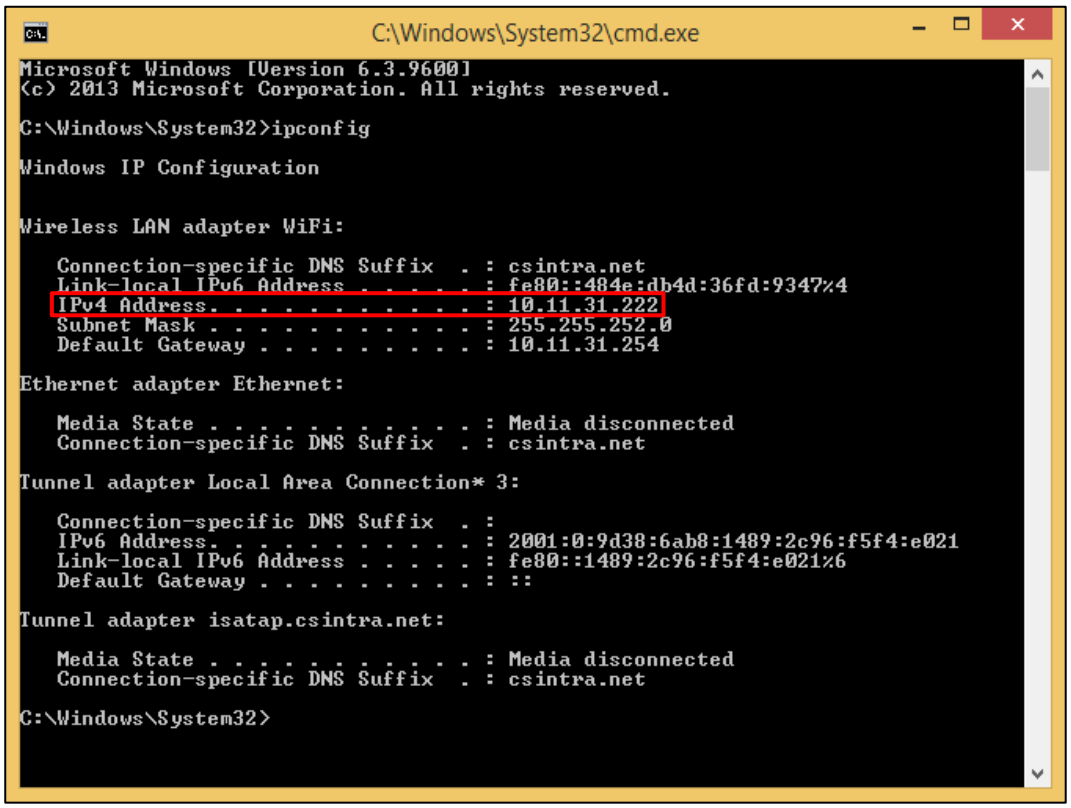


Common Help

Finden der aktuelle IP eines Computers

start -> cmd -> ipconfig

(Example: The Computer is connected via WiFi)



Helpful Links for EJB / Web-Service

http://www.youtube.com/watch?v=9Kf5m7bMu74	Part 1 – Installation
http://www.youtube.com/watch?v=ppGqtOeHm-g	Part 2 – Servlet calling EJB, JSP, timer-EJB
http://www.youtube.com/watch?v=CuLYhkqtoVo	Part 3 – JPA
http://www.http://www.youtube.com/watch?v=1epUpQlKwME	Part 4 Facelets / JSF 2
http://www.youtube.com/watch?v=on55728 9GzA	Part 5
http://www.youtube.com/watch?v=aBjlR9HoR50	Langes vollständiges Beispiel von JSP, Servlet, DB-Connection, JPA, EJB, Web-Services

- SOAP (JAX-WS)

https://www.youtube.com/watch?v=mKjvKPlb1rA	
https://www.youtube.com/watch?v=C53G6R5EVys	
https://www.youtube.com/watch?v=6hqDMS-oJ9k	(Creating stub for a public WSDL)
https://www.youtube.com/watch?v=KFlDdb65w3U	(Call the stub)
https://www.youtube.com/watch?v=Zx6NuO6niFI	(Set up J2EE)
https://www.youtube.com/watch?v=SozdMGNrxro	(Theorie Service first vs contract first)

- REST (JAX-RS)

https://www.youtube.com/watch?v=gKBiIWY7FYw&list=PLu47tUtKqNlwfR-nqjiWUaIWOYEi9FyWo	
https://www.youtube.com/watch?annotation_id=annotation_401609&feature=iv&src_vid=gKBiIWY7FYw&v=4DY46f-LZoM	
https://www.youtube.com/watch?v=xkKcdK1u95s	REST Web Services 01 – Introduction:

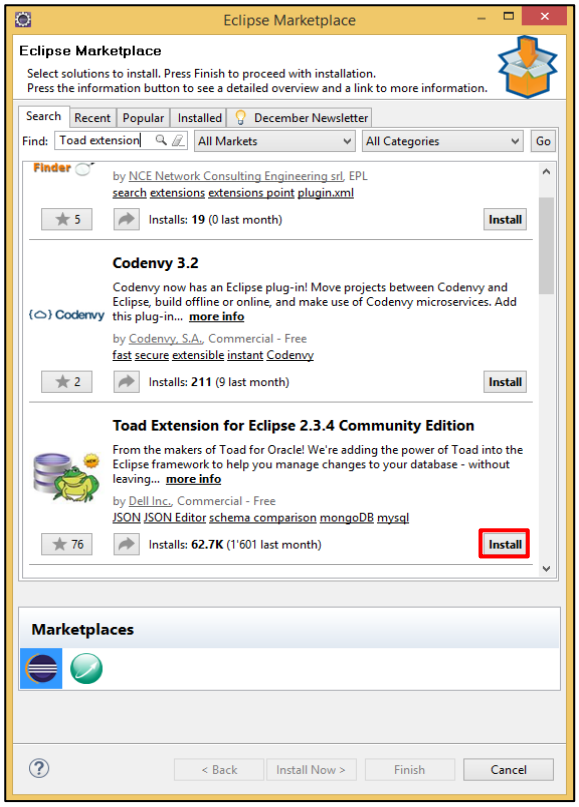
- Web-Service

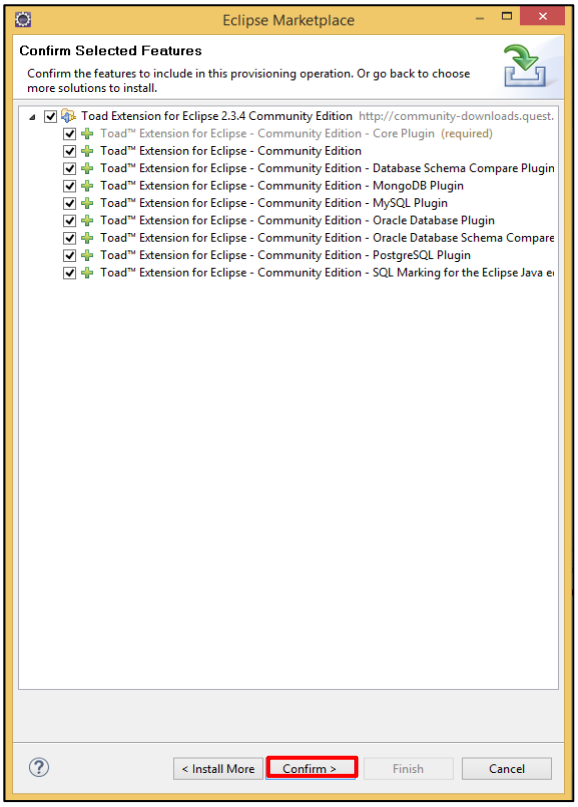
https://www.youtube.com/watch?v=9kboiLpqKY4	
https://www.youtube.com/watch?v=u5cQkVgq6jE	
https://www.youtube.com/watch?v=HLzM92ZnisY	
https://www.youtube.com/watch?v=dbxASThtC_g	Web-Service lecture

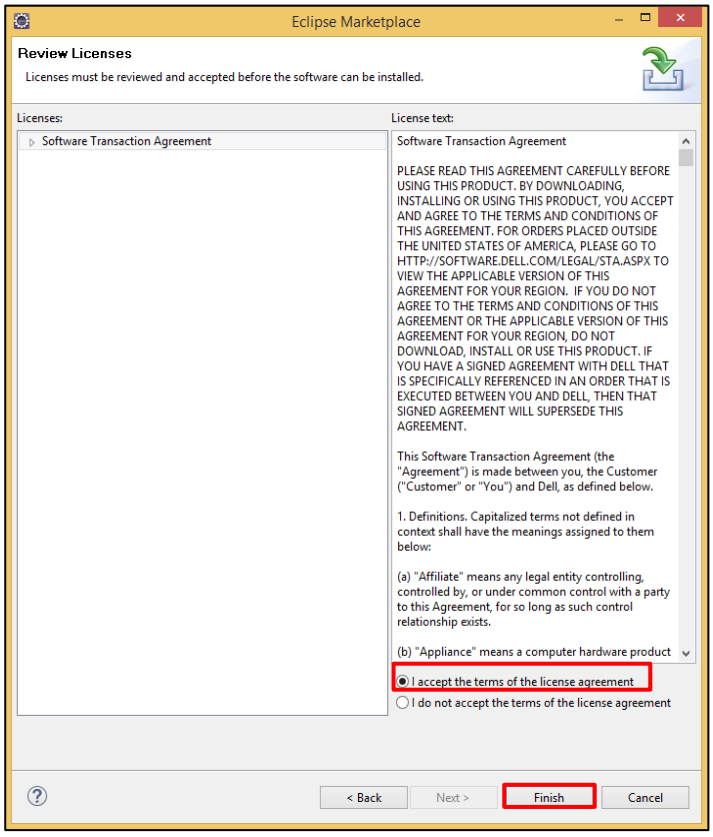
Tips and Tricks for Eclipse

Obsolete: Download and Install MySQL Eclipse Plugin – Toad extension

This part of the guide is obsolete, since Toad Extension for Eclipse is no longer available.
Eclipse -> Help -> Eclipse Marketplace -> Toad Extension

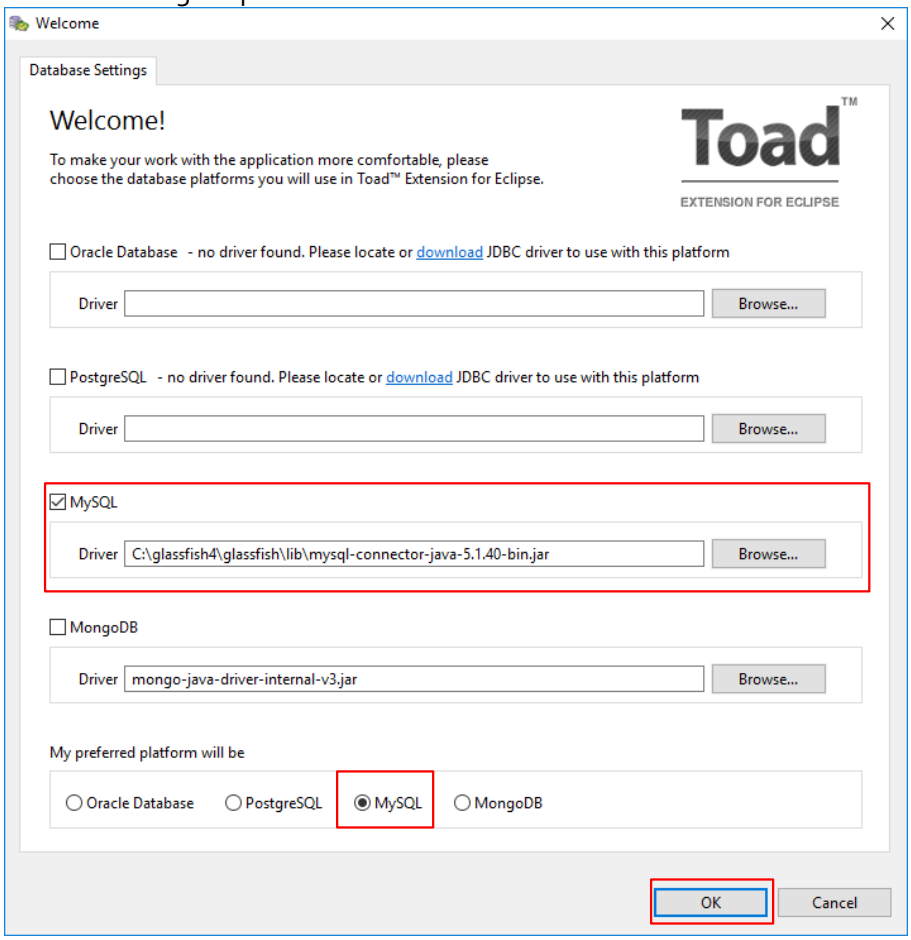






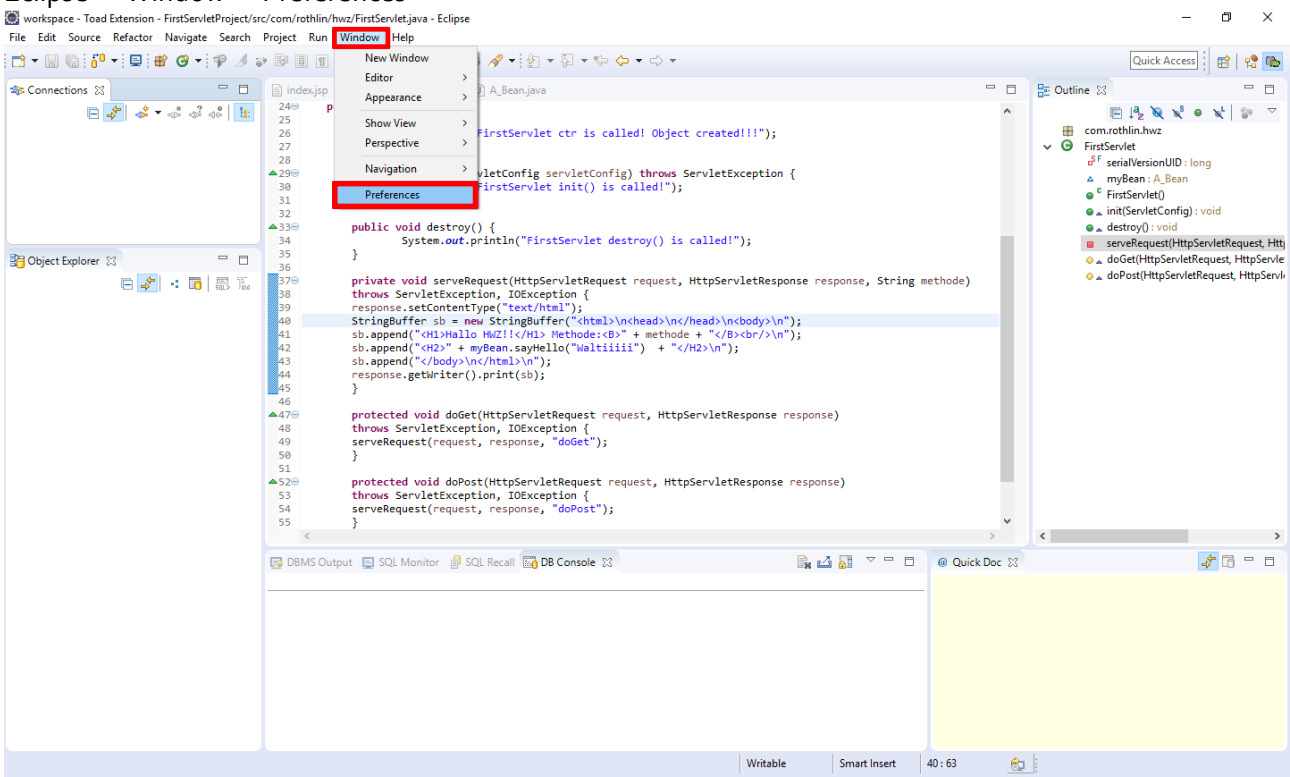
Add MySQL JDBC driver

After restarting Eclipse…

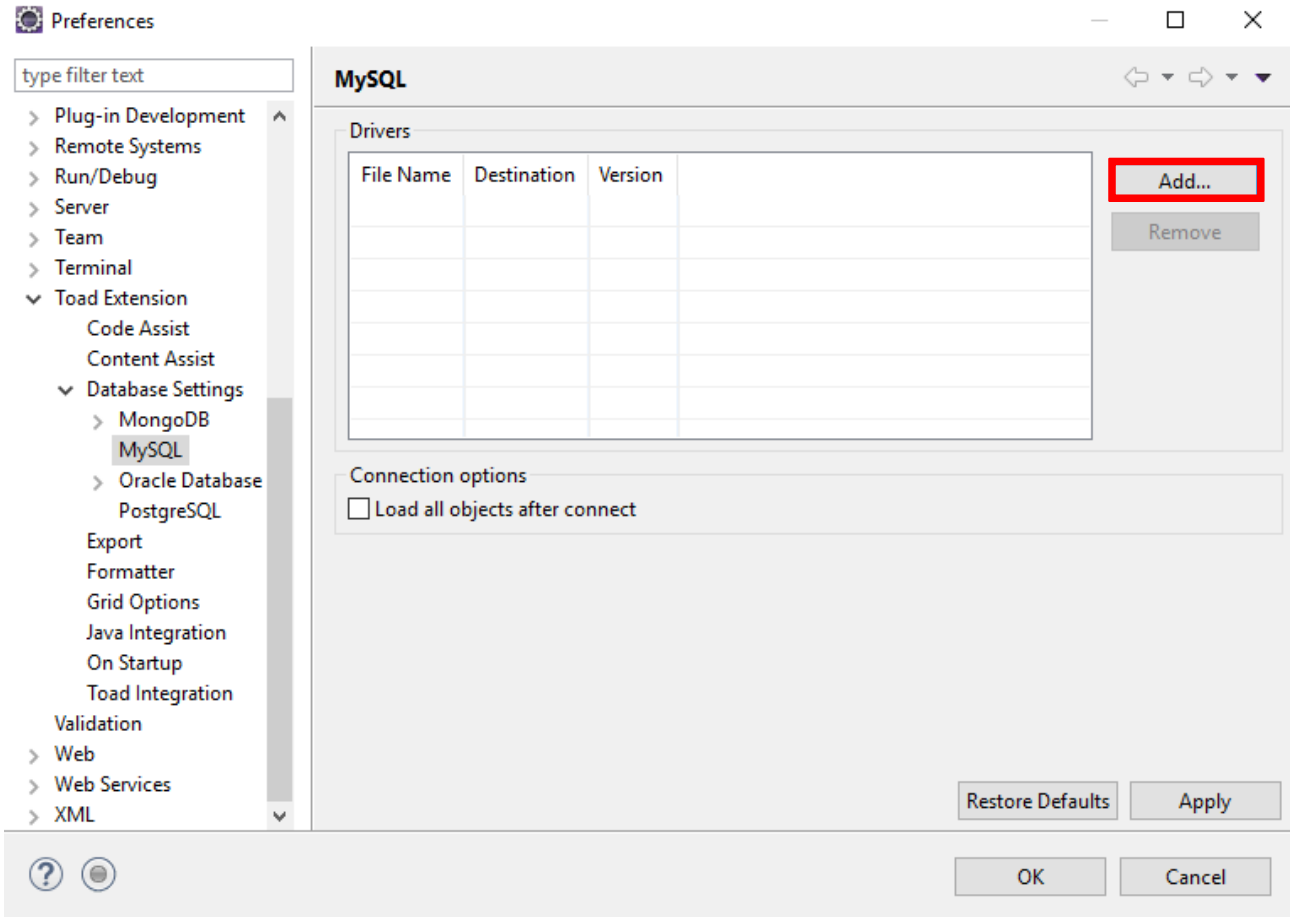


If the window doesn't show up on restart:

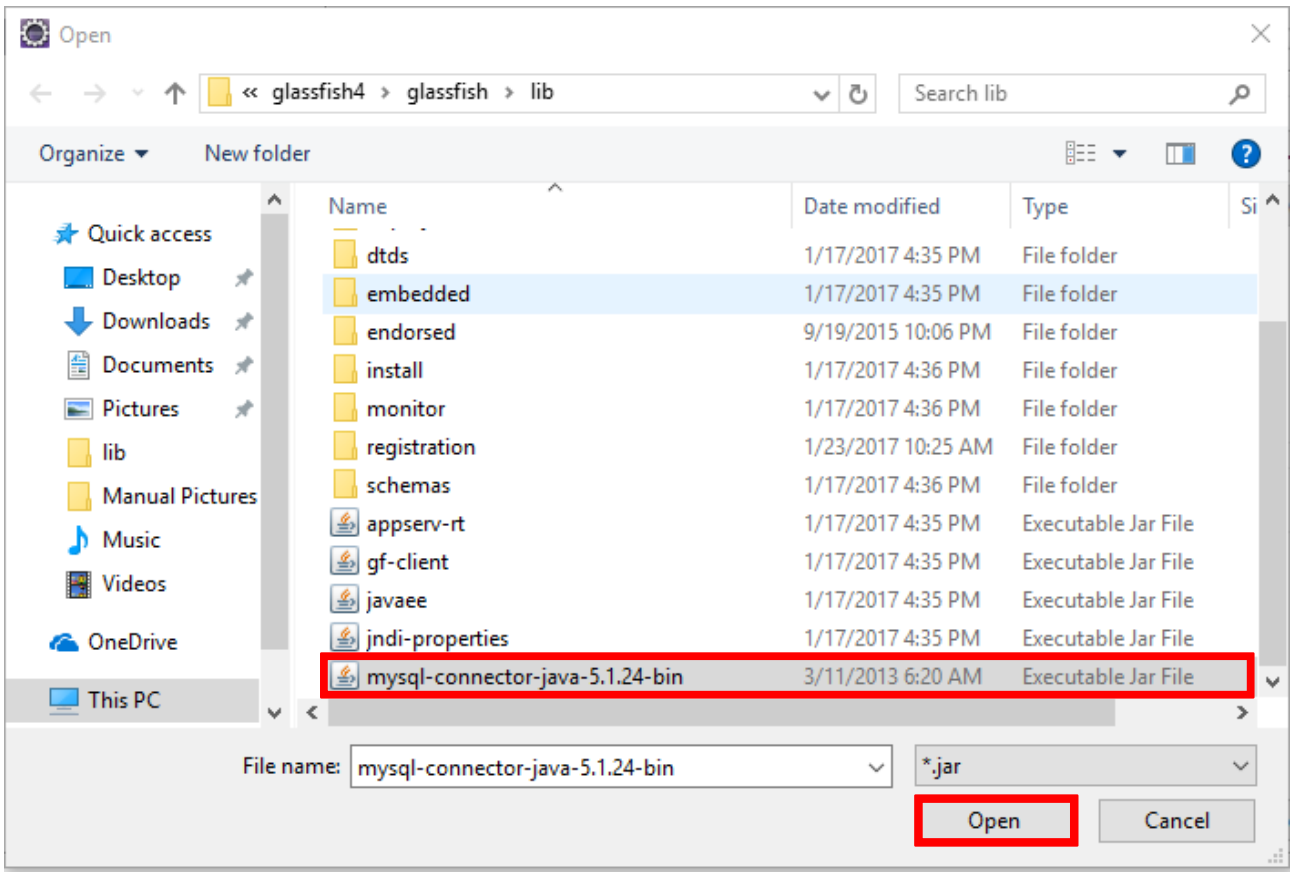
Eclipse -> Window -> Preferences



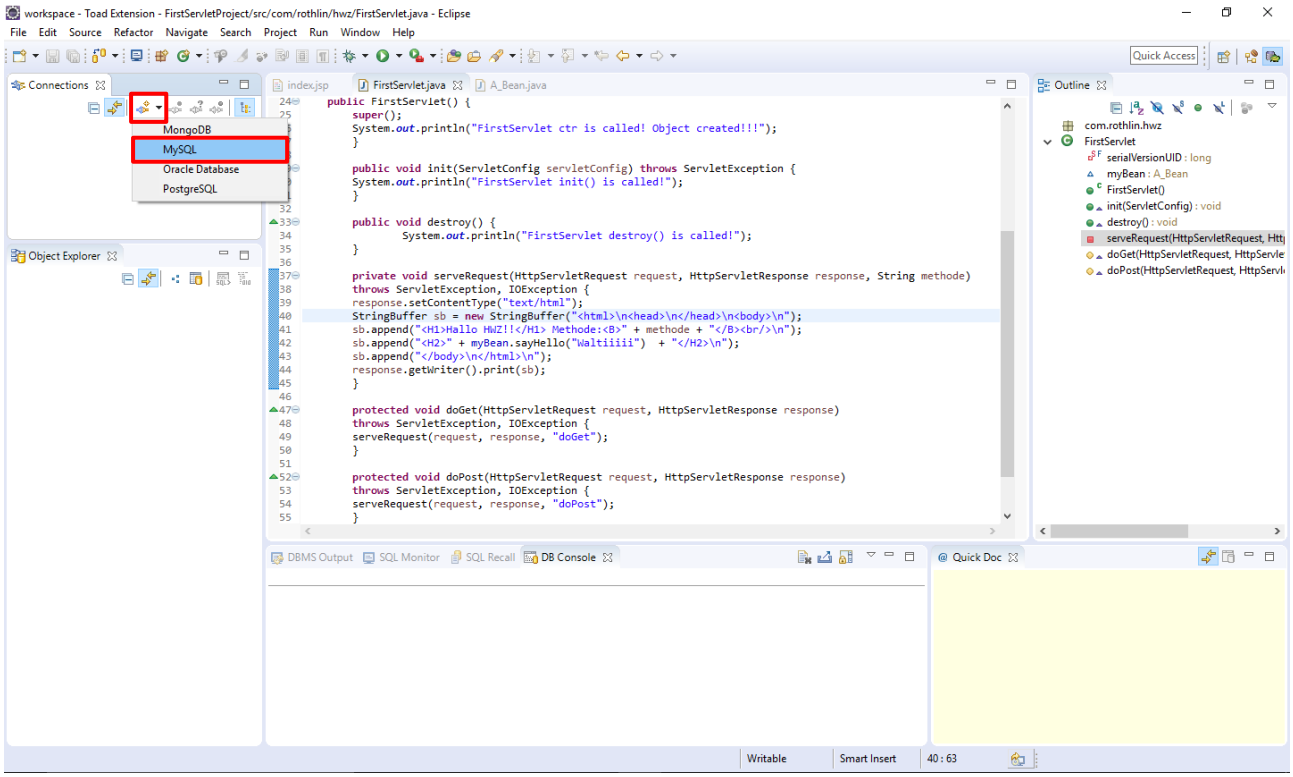
Toad Extension -> Database Settings -> MySQL -> Add



Add the MySQL JDBC driver, which is located at C:\glassfish4\glassfish\lib.



Add MySQL connection



MySQL Root Password: admin

New MySQL Connection

MySQL Connection

Connection is OK.

Manual Settings

Connection Settings | SSL | SSH

Host: localhost

Port: 3306

Database: sakila

Encoding: utf8

MySQL

User: root

Password: *****

Save password

Enable AutoCommit

Restore previous work after login

Connection Name: root@localhost/sakila

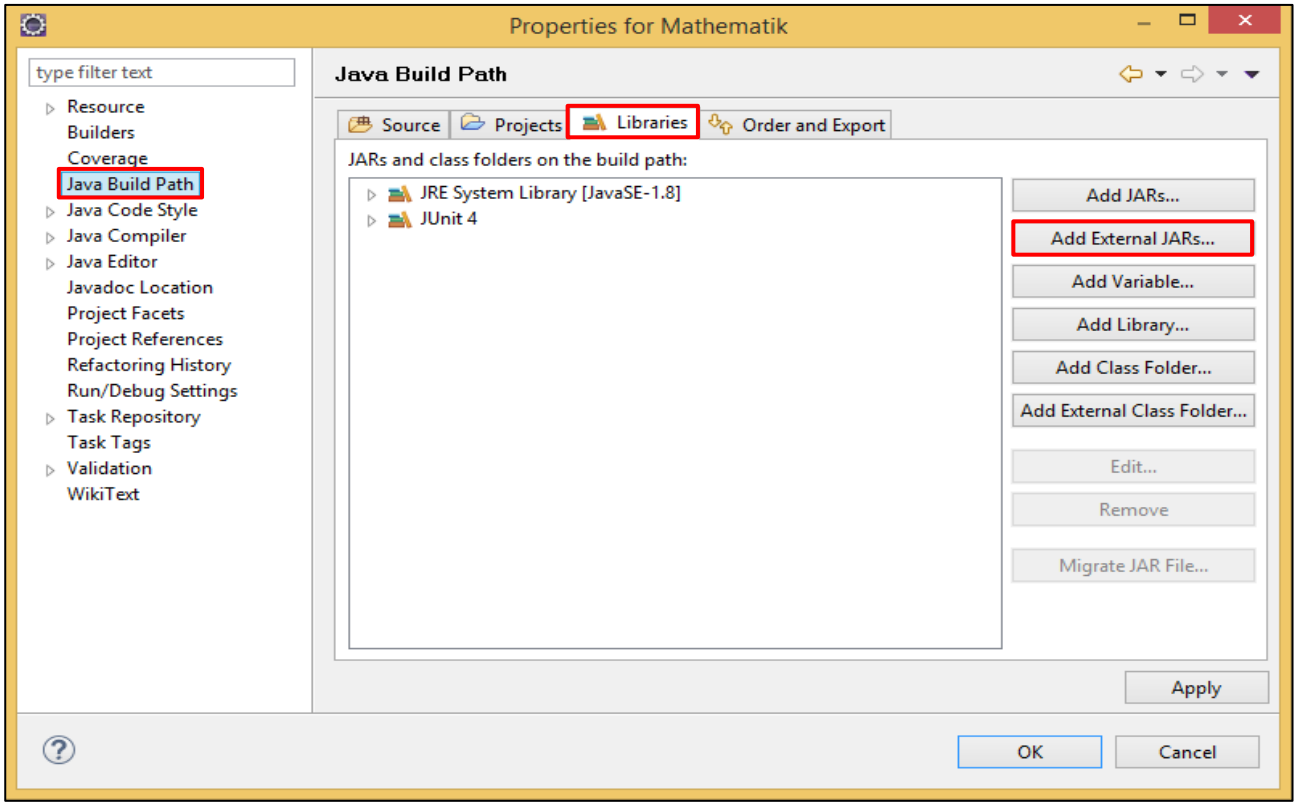
Connection String Preview:

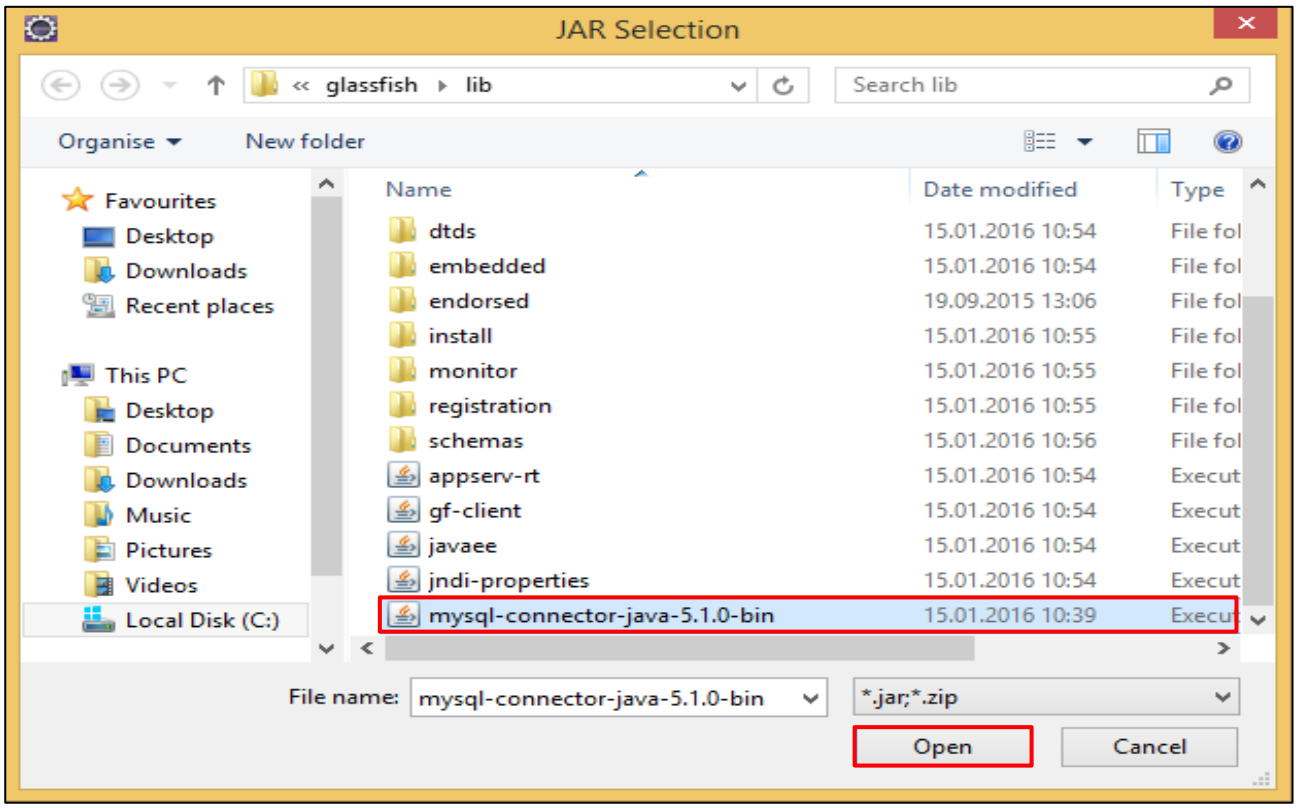
jdbc:mysql://localhost:3306/sakila?characterEncoding=utf8

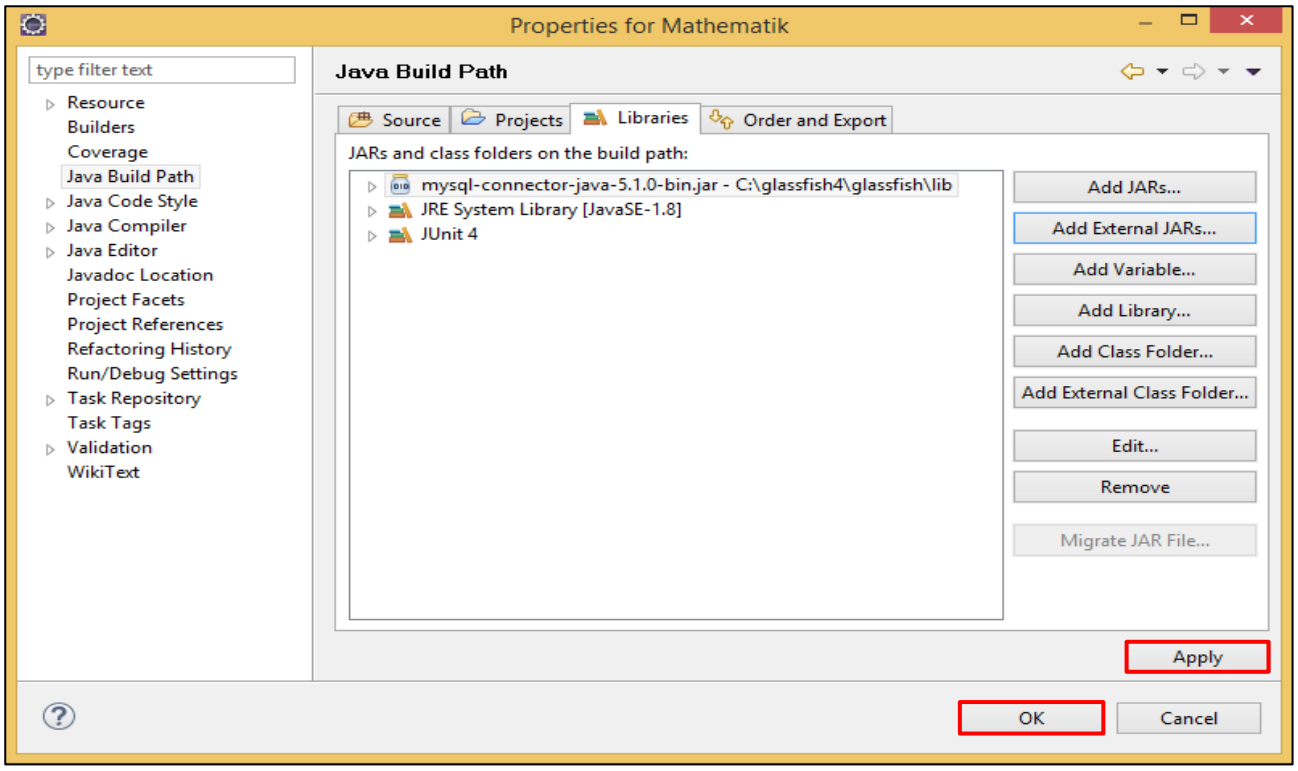
Test Connection | OK | Cancel

New jar to eclipse project

Right click -> Properties -> Java Build Path -> Libraries





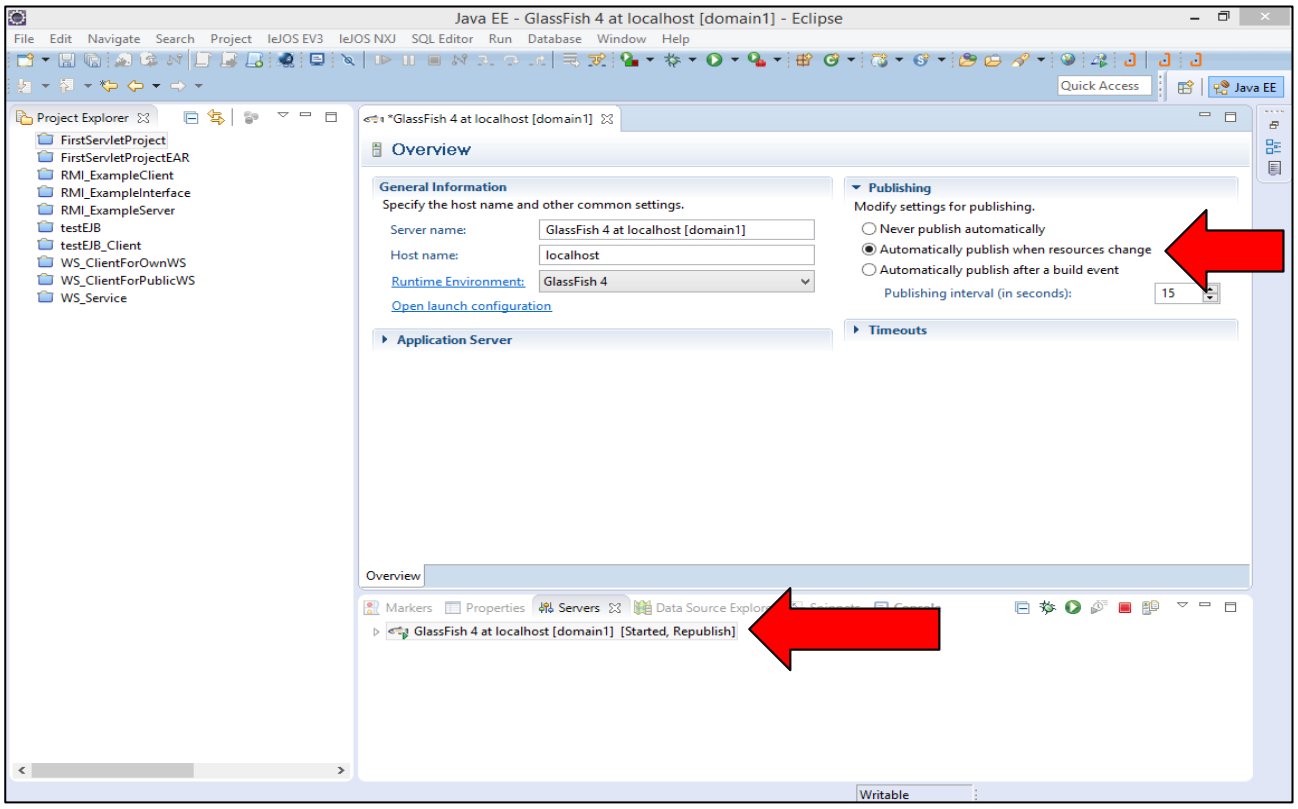


Shortcuts in Eclipse

Combination	Description
`left mousepad`	open declaration
`ctrl + F11`	run application
`ctrl + space`	auto complete
`Type:"sysout" ctrl + space`	system.out.println();
`select string + F2`	show tooltip in box with scrollbar
`shift + F2`	open java doc in browser
`ctrl + shift + O`	Add and removes (un)used imports
`ctrl + shift + F`	format the source code
`ctrl + shift + C`	comment

Automated publishing when source has been saved

Double click on Glassfish 4 at localhost {domain} {Started, Republish}.



Import Project in eclipse

Nach import ist alles rot!!!

Build-Path -> configure Built path
Alle Libraries die gefunden wurden -> remove
Add Library -> JRE System Library -> Workspace default